Simulation und Evaluation künstlicher Blütenbestäubung

Viviane Schöbel

TR 2007-07

(Basierend auf der Diplomarbeit von Viviane Schöbel)

Universität Augsburg
Institut für Informatik

August 2007

© Viviane Schöbel
Institut für Informatik
Universität Augsburg
D-86135 Augsburg, Germany
www.informatik.uni-augsburg.de/vs
- all rights reserved -

Kurzfassung

Einige Bereiche der Informatik sind bereits seit längerer Zeit bestrebt, Phänomene der Natur in künstliche Systeme zu übertragen. Dabei wird unter anderem versucht, das selbst-organisierende Verhalten eines Insektenschwarms künstlich zu erzeugen und effektiv zu nutzen. Ein selbst-organisierendes System in der Biologie ist beispielsweise die Wechselwirkung zwischen Bestäubern und deren Pflanzen. Dieses basiert auf dem Fortpflanzungsdrang einer Pflanze, die für das Fortbestehen ihrer Art Blüten mit Lockmitteln erzeugt, wie auch auf dem Überlebensdrang eines bestäubenden Insektes, das für sein Fortbestehen Nahrung in den Blüten der Pflanzen sucht. Die Symbiose, d.h. die Anstrengungen, die eine Pflanze für eine Bestäubung und ein Bestäuber für das Auffinden von Nahrung auf sich nimmt, sind Bestandteil des Modells der künstlichen Blütenbestäubung, welches die Grundlage dieser Arbeit bildet. Die vorliegende Arbeit umreißt die biologischen Hintergründe und beschreibt das formale Modell der künstlichen Blütenbestäubung. Dieses Modell soll auf Basis einer geeigneten Simulationsumgebung evaluiert werden. Da trotz einer Menge etablierter, bereits bestehender Simulationsumgebungen kein System den Anforderungen des Modells gerecht wird, muss eine neue, für das Modell passende Simulationsumgebung entwickelt werden. Nachdem die für die Evaluation relevanten Kriterien identifiziert sind, wird auf Basis des neu entwickelten Systems eine Bewertung des Modells der künstlichen Blütenbestäubung durchgeführt.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Technische Bereiche, wie z.B. die Informatik, müssen sich, um den Anforderungen des Menschen im Alltags- wie auch Berufsleben gerecht zu werden, kontinuierlich weiter entwickeln. Dabei reicht es nicht aus, nur neue, innovative Ideen umzusetzen. Auch bisher gut etablierte Systeme müssen überarbeitet und verbessert werden. Im Bereich der Informatik existieren z.B. eine Menge stabiler, erfolgreich eingesetzter Software-Architekturen, wobei selbst diese Systeme immer noch einige Probleme aufweisen.

Ein seit langem bestehendes Problem, dessen Lösung im Laufe der Zeit immer dringlicher wird, ist die hohe Komplexität von Software Systemen. Dabei handelt es sich unter anderem um Systeme, die über Jahre hinweg um benötigte Funktionalitäten erweitert wurden und dadurch zu einem Software-"Monstrum" angewachsen sind. Ein solches System besitzt eine enorme Komplexität, die, wenn überhaupt, nur noch sehr schwer von einem Menschen überblickbar und beherrschbar ist. Ein weiteres, teilweise aus der hohen Komplexität resultierendes Problem ist der hohe Kostenfaktor, der sich durch die Wartung eines Software Systems ergibt. Wie bereits bekannt, verschlingt die Wartung eines Systems weit mehr finanzielle Ressourcen als die eigentliche Entwicklung desselben. Je komplexer das System, desto höher sind die Wartungskosten, d.h. die Kosten steigen proportional zu dessen Komplexität. Darüber hinaus ergibt sich zusätzlich das Problem, dass nur noch sehr gut ausgebildete IT-Spezialisten in der Lage sind die bereits sehr hohe Komplexität bestehender Systeme in den Griff zu bekommen. Auf Grund der fachlichen Ausbildung sind diese Fachkräfte wesentlich teurer als normal ausgebildete Arbeitskräfte und erhöhen den Kostenfaktor dadurch zusätzlich.

Die hohe Komplexität ist demnach nicht das einzige, aber das wichtigste Problem, das es zu lösen gilt, da viele weitere Probleme daraus resultieren oder dadurch verstärkt werden. Dieses Vorhaben wird umso dringlicher, da die Komplexität von Systemen im Laufe der nächsten Jahre immer weiter bis zu einem voraussichtlich nicht mehr beherrschbarem Niveau ansteigen wird. Diese Entwicklung lässt sich nicht nur im Bereich der Informatik feststellen. So werden auch Aufgaben in alltäglichen Bereichen immer komplizierter und komplexer. Betrachtet man z.B. den Münchner Hauptbahnhof, so fällt auf, dass hinter einem reibungslosen Ablauf ein enormer Organisationsaufwand steckt. Neben der Koordination ankommender und abfahrender Züge muss des Weiteren besonders flexibel auf eventuell verspätete Züge reagiert werden. Der Organisationsaufwand für einen reibungslosen Ablauf an einem Bahnhof steigt mit der zunehmenden Masse der Bahnkunden immer mehr an,

d.h. die Komplexität des ganzen Systems wird immer größer. Um die Komplexität in Software wie auch alltäglichen Systemen besser zu beherrschen, haben sich aus Industrie wie auch Wissenschaft verschiedene Initiativen gebildet, die neue Problemlösungsansätze erforschen. Diese Forschungsinitiativen lassen sich dabei u.a. durch Phänomene der Natur, wie z.B. die Selbst-Organisation innerhalb einer Insektenkolonie, inspirieren und versuchen mit der künstlichen Nachbildung des aus der Natur gewonnenen Verhaltens die Komplexität eines Systems zu reduzieren.

Die künstliche Blütenbestäubung ist ebenfalls ein aus der Natur inspiriertes Verfahren, dass auf der biologischen Wechselwirkung von Pflanzen und ihren Bestäubern basiert. Dieses Verfahren beruht auf verschiedenen Agententypen (Pflanzen und Bestäuber), die nur in Kooperation miteinander ein gewünschtes Ergebnis erreichen können. So können Blüten nur dann bestäubt werden, wenn es Insekten gibt, die von Blüte zu Blüte fliegen und dabei Pollen zur Bestäubung mit sich tragen. Diese Insekten machen dies jedoch nicht aus Kollegialität den Blüten gegenüber. Vielmehr ist die Bestäubung einer Blüte ein unbeabsichtigter Nebeneffekt, der bei der Nahrungssuche von Insekten wie z.B. Bienen auftritt. Ein Insekt fliegt auf der Suche nach Nektar mehrere Blüten an und nimmt dabei versehentlich Pollen an ihrem haarigen Körper und ihren Beinchen auf. Durch diese Pollen kann die nachfolgende, von dem Insekt angeflogene Blüte bestäubt werden. Hierbei wird das System der Wechselwirkung deutlich, d.h. es kann weder eine Pflanze ohne Insekten, noch eine Insekt ohne Pflanze (inkl. Blüten) überleben. Dieses von der Natur ausgeklügelte System soll nun dazu verwendet werden den Ablauf von komplexen Systemen in der realen Welt zu vereinfachen.

Da für die Idee der künstlichen Blütenbestäubung bisher jedoch nur ein formales Modell und keine praktische Umsetzung vorliegt, wird das künstliche Modell im Rahmen dieser Arbeit nach speziellen Kriterien evaluiert, um einen späteren Einsatz in einer realen Umgebung zu ermöglichen. Besonderes Augenmerk wird dabei auf Skalierbarkeit, Robustheit, Flexibilität und Effizienz gelegt. Die im Rahmen der Evaluation gewonnenen Informationen werden dafür verwendet, das Modell gegebenenfalls anzupassen und für einen realen Einsatz zu optimieren.

Diese Arbeit ist folgendermaßen aufgebaut: In Kapitel 2 werden zu Beginn die für das Verständnis notwendigen Grundlagen im Bereich der Natur-inspirierten Algorithmen vermittelt, um darauf folgend das biologische Vorbild, d.h. die Wechselwirkung zwischen Pflanzen und Bestäubern, vorzustellen. Abschließend wird auf das formale Modell der künstlichen Blütenbestäubung detailliert eingegangen.
Nachdem die Evaluation des Modells ein wichtiger Bestandteil dieser Arbeit wird, schafft Kapitel 3 zu Beginn einen Überblick über relevante Evaluationskriterien. Dabei handelt es sich in erster Linie um die Überlegung, wie Eigenschaften des Modells anhand von Messwerten belegt werden können. Da das Modell noch nicht in einem realen System umgesetzt ist, wird eine passende Simulationsumgebung benötigt, anhand derer die notwendigen Messungen durchgeführt werden können. Für diese Simulations- bzw. Evaluationsumgebung werden in Abschnitt 3.2 notwendige Anforderungen skizziert. Basierend auf diesen Anforderungen wird darauf folgend eine Auswahl an bestehenden Simulationsumgebungen auf ihre Tauglichkeit für die Simulation des formalen Modells geprüft und bewertet.
Kapitel 4 beschäftigt sich mit der Implementierung eines für das Modell passenden Simulators. In diesem Zusammenhang wird in einem ersten Abschnitt die Architektur des Simulators erläutert. Darauf folgend werden die wichtigsten Designentscheidungen erklärt und begründet, bevor in Abschnitt 4.4 Probleme und besondere Erkenntnisse während der Umsetzung des Simulators aufgezeigt

werden.
Für die Evaluierung des Modells werden in Kapitel 5 verschiedene Szenarien entwickelt, anhand derer das Modell in Bezug auf die Evaluationskriterien aus 3.1 evaluiert werden kann. In diesem Zusammenhang wird u.a. die Idee eines fahrerlosen Transportsystems im Bereich der Krankenpflege vorgestellt und eine mögliche Realisierung durch den Einsatz der künstlichen Blütenbestäubung evaluiert.
Abschließend werden in Kapitel 6 die bisherigen Ergebnisse der Evaluation zusammengefasst und mögliche Schwachstellen wie auch Verbesserungsmöglichkeiten des Modells der künstlichen Blütenbestäubung aufgezeigt. Des Weiteren wird ein Ausblick geschaffen, wie die entwickelte Simulationsumgebung verbessert und erweitert werden kann.

Kapitel 2

Künstliche Blütenbestäubung

Das in dieser Arbeit zu evaluierende Modell der künstlichen Blütenbestäubung basiert auf dem biologischen Vorbild der Blütenbestäubung von Samenpflanzen durch nahrungssuchende Bienen. Dieses Kapitel dient dazu einen Überblick über bisherige Forschungsbereiche zu erhalten, die sich mit Algorithmen beschäftigen, welche, wie die künstliche Blütenbestäubung, durch die Biologie bzw. Natur inspiriert sind (siehe Abschnitt 2.1). Bevor die künstliche Blütenbestäubung detailliert erklärt werden kann, wird in Abschnitt 2.2 der biologische Ablauf der natürlichen Bestäubung beschrieben. Dabei werden die wichtigsten Prozesse und Komponenten dieses Vorgangs grundlegend erläutert und nur die im Hinblick auf das künstliche Modell relevanten Details und Probleme detailliert behandelt. Anschließend wird das künstliche Modell der Blütenbestäubung in Abschnitt 2.3 vorgestellt. Dessen Wirkungsweise und Einsatzmöglichkeiten werden durch ein anschauliches Szenario in Kapitel 5 verdeutlicht.

2.1 Grundlagen

Das Modell der künstlichen Blütenbestäubung ist, auf den ersten Blick betrachtet, ein weiterer durch die Biologie inspirierter Ansatz, der das selbst-organisierende Verhalten von Insekten in der Natur zum Vorbild nimmt. Die Blütenbestäubung grenzt sich jedoch grundlegend von bisherigen Ideen ab. Alte wie auch aktuelle Ansätze orientieren sich in erster Linie an dem Verhalten einer Menge gleicher Individuen einer Art (z.B. Bienen). Diese kommunizieren untereinander, ohne dass ein Individuum einer anderen Art (z.B. eine Ameise) das Modell relevant beeinflusst.

Die künstliche Blütenbestäubung hingegen orientiert sich an der Wechselwirkung zwischen Individuen zweier Arten. Dabei handelt es sich um die gegenseitige Abhängigkeit zwischen Pflanzen und deren Bestäubern, wobei die einzelnen Pflanzen untereinander wie auch die Bestäuber untereinander nicht miteinander interagieren. Einzig zwischen den Individuen der verschiedenen Arten, d.h. zwischen einer Pflanze und deren Bestäuber, kommt eine Interaktion zustande.

Die für die Wechselwirkung notwendigen Bestäuber sind unter anderem Insekten aus Insektenschwärmen, wie z.B. Bienen. Das Studium von Insektenschwärmen wie auch anderen Phänomenen aus der Natur wird u.a. in den Bereichen der Schwarmintelligenz, des Autonomic Computing wie auch des Organic Computing betrieben. In diesem Zusammenhang sind die beiden Begriffe “Selbst-Organisation” und “Emergenz” für ein Verständnis des Verhaltens eines Insektenschwarms

unerlässlich. Es gibt eine Vielzahl an Definitionen für beide Begriffe, so dass in den beiden folgenden Abschnitten ein kurzer Überblick über die in dieser Arbeit verwendeten Definitionen gegeben wird, bevor die Bereiche der Schwarmintelligenz und des Autonomic wie auch Organic Computing dem Leser vorgestellt werden.

2.1.1 Selbst-Organisation

Der Begriff der Selbst-Organisation beschreibt die Entstehung einer inneren Struktur aus einer völlig chaotischen Menge heraus. Das beste Beispiel für ein selbst-organisierendes Verhalten sind Ameisen auf ihrer Nahrungssuche. Dabei suchen Ameisen willkürlich in der Umgebung nach Nahrung. Ist eine Ameise auf eine Nahrungsquelle gestoßen, so lässt sie auf dem Weg zurück zu ihrem Ameisenhaufen so genannte Pheromone, d.h. Duftstoffe, auf dem Boden zurück. Indem eine andere Ameise dieser Pheromonspur folgt, kann sie ebenfalls die Nahrungsquelle finden. Dadurch bildet sich aus einer unkoordinierten Menge an willkürlich suchenden Ameisen die Struktur einer Ameisenstrasse, da immer mehr Ameisen auf die Duftstoffe stoßen, diesen bis zu der Nahrungsquelle folgen und die geladenen Nahrung zu ihrem Ameisenhaufen bringen. Bemerkenswert an diesem Verhalten ist der Umstand, dass obwohl jede Ameise lokal einem eigenen Plan folgt (die Nahrungssuche), die Struktur der Ameisenstrasse bzw. das globale Verhalten des Schwarms nicht durch einen zentralen Plan, eine "Führerameise" oder durch Steuerung von außen gebildet wurde, sondern selbst-organisierend durch die Interaktion der Ameisen untereinander entstanden ist.

Eine einheitliche Definition des Begriffs Selbst-Organisation gibt es in der Literatur jedoch nicht. De Wolf und Holvoet [9] sehen Selbst-Organisation als einen dynamischen, adaptiven Prozess, in dem Systeme ohne externe Kontrolle eine innere Struktur selbst erlangen und beibehalten. Eine ähnliche Definition von Selbst-Organisation findet sich in [1] wieder. Demnach ist ein System selbst-organisierend, falls es sich selbstständig verändert als durch eine externe Instanz verändert zu werden. Laut [6] hingegen bilden sich in einem selbst-organisierenden System Muster auf globaler Ebene, die ihren Ursprung ausschließlich aus Interaktionen zwischen Komponenten niedrigerer Ebene haben.

Wie zu sehen ist, gibt es eine Vielzahl an verschiedenen Definitionen für ein selbst-organisierendes System. Die in dieser Arbeit verwendete Definition ist von de Wolf und Holvoet.

2.1.2 Emergenz

Der Begriff der Emergenz steht in starkem Zusammenhang mit der Selbst-Organisation einer Menge an Individuen. Wie eben beschrieben, besitzt ein selbst-organisierendes System innere Strukturen, die durch Interaktion einzelner Individuen entstehen, die eine bestimmte Aufgabe versuchen zu erfüllen. Die entstandenen Strukturen sind von dem lokalen Verhalten der einzelnen Individuen abhängig, können jedoch meist nicht direkt mit ihnen in Verbindung gebracht werden. Das globale Verhalten eines Systems besteht demnach nicht nur aus den Aktionen der einzelnen Individuen, sondern besitzt auch unvorhersehbare Eigenschaften, die sich auf Grund der Interaktion zwischen den einzelnen Individuen bilden. In der Wisschenschaft wird dieses Phänomen als Emergenz bezeichnet. Wann ein System jedoch eine emergente Eigenschaft besitzt, wurde in einer Vielzahl verschiedener, sich teilweise gegenseitig ausschließender Definitionen versucht festzulegen. Für de Wolf [8] beispielsweise, weist ein System Emergenz auf, wenn auf Systemebene Strukturen bzw.

Eigenschaften dynamisch entstehen, die ihren Ursprung in den Interaktionen der einzelnen Systemkomponenten haben. In [24] hingegen, wird Emergenz als Resultat kollektiver Selbst-Organisation gesehen, da interessante Eigenschaften auf Systemebene durch die Interaktion identischer oder sehr ähnlich aufgebauter Komponenten erzeugt werden. Des Weiteren gibt es einige detailliertere Definitionen des Begriffs aus dem Bereich der Philosophie des Geistes. Laut Stephan [23] wird der Begriff Emergenz in schwache Emergenz und starke Emergenz aufgeteilt. Schwache Emergenz legt die Basiseigenschaften von emergentem Verhalten fest, starke Emergenz hingegen erweitert diese Eigenschaften, um die Menge an emergenten Eigenschaften stärker einzuschränken.

Schwache Emergenz Schwache Emergenz definiert die grundlegenden Voraussetzungen, die eine Eigenschaft, Struktur oder ein Verhalten eines Systems besitzen muss, um als emergent zu gelten. Demnach werden schwach emergente Eigenschaften durch die folgenden drei Merkmale identifiziert:

- Eine emergente Eigenschaft, Anordnung, Struktur oder emergentes Verhalten entsteht nur in Systemen, die ausschließlich aus physischen Komponenten bestehen. Demnach werden u.a. übernatürliche Phänomene nicht als emergente Eigenschaften angesehen.
- Die Eigenschaften eines Systems lassen sich in zwei Klassen aufteilen. So kann ein System eine Eigenschaft besitzen, die ebenfalls in einer Systemkomponente besteht oder aber in keinem Teil des Systems zu finden ist. Eigenschaften, für die letzteres gilt, sind Kandidaten für emergente Eigenschaften.
- Die Eigenschaften wie auch das Verhalten eines Systems hängt von dessen Mikrostruktur ab. Emergente Eigenschaften eines Systems können sich nur dann verändern, wenn dessen Mikrostruktur geändert wird.

Anhand dieser Merkmale können schwach emergente Systemeigenschaften identifiziert werden. Die Menge dieser Eigenschaften beinhaltet jedoch noch eine Vielzahl an uninteressanten Eigenschaften, so dass sie durch eine Definition der stark emergenten Eigenschaften weiter eingeschränkt werden kann.

Starke Emergenz Die starke Emergenz basiert auf den Merkmalen der schwachen Emergenz und schränkt deren Ergebnisraum weiter ein. Demnach ist eine stark emergente Eigenschaft laut Stephan dadurch definiert, dass sie die Merkmale der schwachen Emergenz erfüllen muss und zusätzlich nicht reduzierbar sein darf. Ein Systemeigenschaft ist nicht reduzierbar, wenn sie

1. nicht durch das Verhalten, die Eigenschaften oder die Struktur der Systemkomponenten entsteht und
2. nicht durch Eigenschaften entsteht, die Systemkomponenten isoliert oder innerhalb einer anderen Konfiguration aufweisen.

Eine Systemeigenschaft ist demnach stark emergent, wenn sie schwach emergent und nicht auf eine Eigenschaft einer Systemkomponente reduzierbar ist.

Auf Grund der Vielzahl an Definitionen des Begriffs Emergenz, muss für ein einheitliches Verständnis dieser Arbeit eine geeignete Definition gewählt werden. Da das Modell der künstlichen

Blütenbestäubung auf der Selbst-Organisation nahrungssuchender Bienen basiert und in dieser Arbeit u.a. die emergenten Eigenschaften des Systems betrachtet werden sollen, ist die Definition von de Wolf für diese Arbeit relevant.

2.1.3 Schwarmintelligenz

Der Begriff der Schwarmintelligenz (SI) [3], ein Bereich der verteilten künstlichen Intelligenz, wurde 1989 von G.Beni und J.Wang im Bereich der Robotikforschung geprägt. Dieser Bereich beschäftigt sich in erster Linie mit der Entwicklung neuer Algorithmen, die durch das selbst-organisierende Verhalten von Insektenschwärmen inspiriert sind. Ein Schwarm besteht aus einer Vielzahl einzelner, gleichartiger Individuen, die alle erstrebt sind dieselben Aufgaben nach einem festen Muster zu erfüllen. Dafür müssen die Individuen untereinander und mit ihrer Umwelt kommunizieren bzw. interagieren, was u.a. durch die Abgabe von Pheromonen und Duftstoffen erzielt werden kann. Ein Schwarm kann z.B. ein Ameisen-, Bienen- oder anderer Insektenstaat sein. Die Schwarmintelligenz ist bemüht die aus dem kollektiven Verhalten von Schwärmen gewonnenen Erkenntnisse für die Lösung von Optimierungsproblemen einzusetzen.

2.1.4 Autonomic Computing

In Zusammenhang mit dem von IBM 2001 geprägten Begriff des Autonomic Computing (AC)[14] werden autonome, d.h. selbst-verwaltende Systeme erforscht. Das künstliche Selbst-Management orientiert sich hierbei an der natürlichen Selbst-Verwaltung des vegetativen Nervensystems des Menschen. Auf Basis dessen wurden vier Zielsetzungen für AC Systeme formuliert: Selbst-Konfiguration, Selbst-Optimierung, Selbst-Schutz und Selbst-Heilung. Um diese Zielsetzungen erfüllen zu können, muss das System bzw. die autonome Komponente einerseits in der Lage sein sich selbst zu überwachen und zu regeln und sich andererseits seines Zustands wie auch des Zustands seiner Umwelt bewusst sein. Für die Umsetzung dieser Eigenschaften hat IBM das Konzept des Autonomic Managers und eine passende Referenzarchitektur dazu (nachzulesen in [15]) entwickelt. Zusammengefasst ist ein AC System ein selbst-verwaltendes System, dessen Organisation nicht wie in Abschnitt 2.1.3 verteilt auf verschiedene Individuen abläuft, sondern von einer oder mehreren zentralen Instanzen geregelt wird.

2.1.5 Organic Computing

Organic Computing(OC) ist wie SI und AC ein weiterer von der Natur inspirierter Forschungsbereich. Wie der Name schon sagt, handelt es sich bei Organic Computing um die Idee eines ‘Organischen Systems’, das lebensähnlich aufgebaut ist und lebensähnlich handeln soll. Laut Müller-Schloer ist OC folgendermaßen definiert [19]:

> Ein ‘organischer Computer’(OC) ist definiert als ein selbst-organisierendes System, das sich den jeweiligen Umgebungsbedürfnissen dynamisch anpasst.

Nach Müller-Schloer sollte ein Organic Computing System so genannte Selbst-x Eigenschaften besitzen, d.h. es sollte selbst-schützend, selbst-heilend, selbst-konfigurierend, selbst-optimierend und selbst-erklärend sein. Auch wenn der Großteil dieser Eigenschaften auch in AC Systemen benötigt

wird, grenzt sich OC dennoch dahingehend ab, dass emergentes Verhalten innerhalb des Systems explizit berücksichtigt, d.h. auch verteilte selbst-organisierende Systeme ohne zentrale Instanz möglich sind. Des Weiteren grenzen sich die beiden Gebiete durch ihren Forschungshintergrund und ihre Einsatzgebiet voneinander ab. Demnach werden OC-Systeme in erster Linie für eingebettete und ubiquitäre Systeme entwickelt. AC hingegen wurde in erster Linie für große Serverarchitekturen entwickelt. Da OC Ansätze aus der Schwarmintelligenz verwendet, aber auch, wie in [22] beschrieben, selbst-organisierende Systeme mit einer oder mehreren zentralen Instanzen berücksichtigt, kann der Bereich des OC als Erweiterung der SI gesehen werden.

2.2 Biologisches Vorbild

Das zu evaluierende Modell orientiert sich an der Blütenbestäubung von Samenpflanzen, die, wie z.B. die Hagebutte, Blüten zur Befruchtung hervorbringen, um nach einer erfolgreichen Befruchtung Samen für die Entwicklung neuer Pflanzen zu bilden.

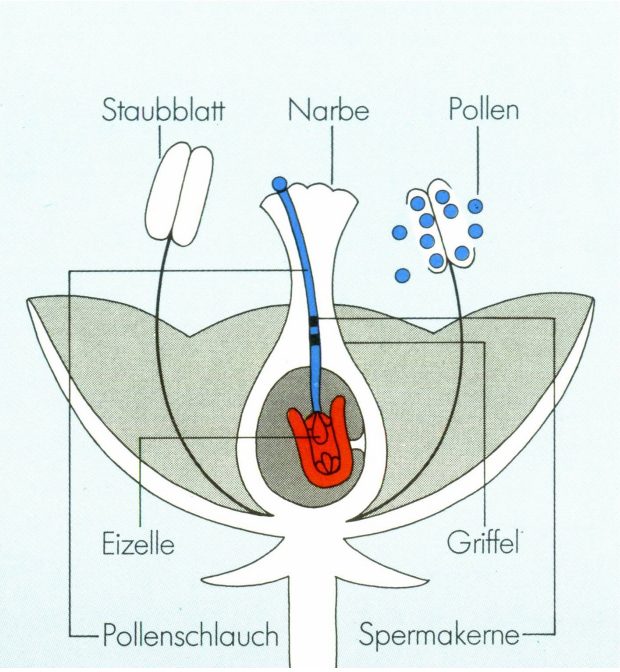

Abbildung 2.1: Schematischer Längsschnitt durch eine zwittrige Blume mit männlichen und weiblichen Blütenteilen. [2]

Bei der Blütenbestäubung handelt es sich um einen in der Biologie unerlässlichen Mechanismus, der den Fortbestand der Pflanzenvielfalt gewährleistet. Um eine Blüte zu bestäuben, müssen die auf den Staubblättern der Blüte liegenden Pollen, d.h. die männlichen Gametophyten, auf die Narbe des Fruchtblatts einer gleichartigen Blüte[1] transportiert werden (siehe Abbildung 2.1). Sobald

[1] Eine gleichartige Blüte ist entweder genau dieselbe oder aber eine in Aufbau und Merkmalen bzw. Art und Gattung identische Blüte.

ein Pollenkorn auf der Narbe liegt, bildet es einen Pollenschlauch, der durch das Fruchtblatt bis zu den Samenanlagen wächst und dadurch eine Befruchtung zwischen männlichen Gametophyten und weiblichem Gametophyt ermöglicht. Die Blütenbestäubung ist demnach unerlässlich für die Befruchtung einer Blüte.

Obwohl die Umsetzung des bisher beschriebene Vorgangs sehr einfach erscheint, steckt ein sehr kompliziertes, von der Natur ausgeklügeltes System hinter einer erfolgreichen Blütenbestäubung. Die einzelnen Komponenten wie auch der grundlegende Ablauf bzw. Aufbau des Systems werden im Folgenden genauer beleuchtet.

2.2.1 Komponenten der Blütenbestäubung

Das System der Blütenbestäubung besteht im Grunde aus zwei Komponenten, den Samenpflanzen und den Pollenträgern.

Die Samenpflanzen bilden Blüten aus, welche die Bestäubungsquelle, d.h. die Pollen auf den Staubblättern, wie auch die Bestäubungssenke, d.h. die Narbe des Fruchtblatts, zur Verfügung stellen. Um Pollenträger für eine Bestäubung anzulocken entwickeln Blüten einen eindeutige Duft, den sie zu bestimmten Tages- oder Nachtzeiten entfalten. Desweiteren haben sie verschiedenfarbige, auffallend geformte Kelchblätter, die ebenfalls der Anziehung von potentiellen Pollenträgern dienen. Die Ausprägung der Kelchblätter wie auch des Duftes hängen von der Gattung und der Art einer Pflanze ab. Nimmt man z.B. die Gattung der Rose, so existiert eine Vielzahl an Rosenarten von der 'Englischen Rose' über die 'Kletterrose' bis hin zur 'Wildrose', die allesamt unterschiedliche optische und geruchliche Merkmale haben.

Die zweite wichtige Komponente im System der Blütenbestäubung ist die Klasse der Pollenträger, auch Bestäuber genannt. Diese befördern die Pollen von den Staubblättern auf die Narbe des Fruchtblatts. Als Pollenträger können verschiedenste Tiere wie z.B. Vögel, Fledermäuse und Insekten, aber auch Wind und Wasser zum Einsatz kommen, wobei die in der Natur erfolgreichsten Bestäuber Insekten wie z.B. Bienen, Hummeln, Käfer, Fliegen, Falter und Schmetterlinge sind. Im Folgenden werden nur noch (fliegende) Insekten als Bestäuber genauer betrachtet, da eine Bestäubung durch Wind und Wasser in der künstliche Blütenbestäubung außer Acht gelassen wird.

2.2.2 Ablauf der Blütenbestäubung

Das große Phänomen der Blütenbestäubung liegt in der Wechselwirkung zwischen Bestäuber und Pflanze. Eine Pflanze bietet Blüten mit Pollen und Narbe an und erhofft sich eine Bestäubung durch einen Pollenträger. Dieser fliegt allerdings nicht von Blüte zu Blüte der Bestäubung wegen, sondern ist in erster Linie auf der Suche nach Nahrung in Form von Nektar. Durch Duft, Farbe und Form der Blüte wird der Pollenträger angezogen und durch den Nektar bei einem Anflug auf die Blüte belohnt. Das Verhalten einer Blüte kann in die folgenden Teilprozesse gegliedert werden.

Die Anziehung: Wie bereits erwähnt versucht eine Blüte einen Pollenträger durch ihren Duft und die Form und Farbe ihrer Kelchblätter anzulocken. Diese Merkmale lassen sich in optische und geruchliche Reizmittel für die Sinnesorgane der bestäubenden Insekten unterteilen und bestimmen welche Pollenträger angelockt werden. So bevorzugen Käfer laut [10] überwiegend weiße Blüten mit vielen Pollen wie z.B. die Seerose, da sie nur Pollen und keinen Nektar aufnehmen können. Nektarsuchende Bienen und Hummeln orientieren sich an farblich sehr

auffälligen Blüten wie z.B. der Zitronenmelisse oder dem Klee, die eine deutliche Lippe zur Landung der Insekten zur Verfügung stellen. Desweiteren muss der Nektar der Blüte in einer kurzen Röhre oder einem kurzen Sporn haften, da Bienen und Hummeln im Gegensatz zu Faltern kürzere Rüssel zu dessen Aufnahme besitzen. Neben den optischen Reizen bieten die meisten Blüten auch geruchliche Reize, die es den bestäubenden Insekten ermöglichen die Blüten der passenden Pflanzenarten aus einer Vielzahl an Blüten einfacher auszuwählen. Wie bei den optischen Merkmalen werden auch mit den geruchlichen Merkmalen nur bestimmte Pollenträger angezogen. So geben z.B. auf Fliegen spezialisierte Blüten ein für die menschliche Nase übelriechendes Aroma ab.

Die Belohnung: Die Anziehungskraft einer Blüte auf bestimmte Pollenträger liegt in ihren Merkmalen, wobei nicht die Schönheit der Farbe oder der angenehme Duft die Insekten anlockt, sondern das Wissen über die Existenz von Nektar. Dementsprechend werden den Insekten durch die Blütenmerkmale implizit Informationen über die Blüte übermittelt, d.h. die Insekten wissen, welche Blüten für ihre Bedürfnisse geeignet sind und welche ihnen nicht genug Pollen oder Nektar zur Verfügung stellen können. Sollte sich ein Insekt von den Merkmalen einer Blüte in die Irre führen lassen, so sind z.B. Bienen und Hummeln in der Lage dies zu registrieren und Blüten mit den gleichen Merkmalen nicht mehr bevorzugt anzufliegen, d.h. sich an die geänderte Umgebung anzupassen. Die Belohnung, die die Merkmale einer Blüte versprechen, sind unter anderem Nektar und Pollen. Nektar ist eine reiche Energiequelle für Insekten, d.h. er wird in erster Linie konsumiert um ein Fortbewegen, z.B. den Flug von Blüte zu Blüte, zu ermöglichen. Des Weiteren stellen z.B. Bienen aus dem Überschuss des gesammelten Nektars Honig her, der für den Winter im Bienenstock bevorratet wird. Pollen hingegen sind für Pflanzen wesentlich aufwändiger zu produzieren als Nektar, da sie in erster Linie aus Eiweiß bestehen, vitaminreicher und mineralhaltiger sind. Dementsprechend sind Pollen für Insekten eine entscheidende Überlebenskomponente, da sie die wohl wichtigste Eiweißzufuhr für diese bieten.

Die meisten Insekten haben ein bestimmtes Verhaltens- bzw. Sammelmuster bei der Nahrungssuche. So fliegen Bienen, die eine Blüte mit reichlich Nektar gefunden haben nur noch Blüten der gleichen Art in ihrer Umgebung an, da diese Blüten mit hoher Wahrscheinlichkeit ebenso reichhaltige Nektarquellen sind. Dadurch wird auch die Wahrscheinlichkeit einer erfolgreichen Bestäubung enorm erhöht, da Pollen und Narbe für eine Bestäubung von einer gleichartigen Blüte stammen müssen. Falls eine Biene keine weiteren Blüten mit den gleichen Merkmalen findet oder aber mit Nektar voll beladen ist, kehrt sie zu ihrem Bienenstock zurück, lädt ihre Fracht ab und geht erneut auf Suche nach Nektar.

2.2.3 Selbstbestäubung versus Fremdbestäubung

Bisher wurde die Bestäubung als Übertragung von Pollen auf die Narbe einer Blüte angesehen. Dabei wurde keine Unterscheidung getroffen, ob die Pollen und die Narbe von derselben oder von unterschiedlichen Blüten stammen. Sollte eine Blüte durch Pollen von sich selbst bestäubt werden, so nennt man dies Selbstbestäubung bzw. Autogamie. Die Bestäubung einer Blüte durch Pollen einer anderen Blüte wird hingegen Fremdbestäubung bzw. Allogamie genannt. In der Natur ist eine Selbstbestäubung nur selten erwünscht, da dadurch die natürliche Mischung des Genmaterials

zweier Pflanzen verhindert wird, die selbstbestäubte Pflanze ihre Anpassungsfähigkeit verliert und auf Grund dessen im Laufe der Jahre eventuell ausstirbt. Trotz allem sind einige Pflanzenarten auf Selbstbestäubung angewiesen, da deren Lebensraum beispielsweise sehr extremen Temperaturen unterliegt, so dass nur sehr wenige bis keine Bestäuber überleben können. Der Großteil der Samenpflanzen versucht sich allerdings durch Fremdbestäubung zu vermehren, was einen Ausschluss der Selbstbestäubung notwendig macht. Dementsprechend kann man in der Natur einige Anpassungen erkennen, die durch räumliche oder zeitliche Trennung eine Selbstbestäubung verhindern sollen.

- **Zeitliche Trennung**
  - **Vormännlichkeit:** Bei der Vormännlichkeit entleert eine Blüte erst ihre Staubbeutel an den Staubblättern, bevor die Narbe der Blüte empfängnisbereit ist. Dadurch können zwar Pollen einer Blüte auf die Narbe derselben Blüte kommen, allerdings hat dies keine Bestäubung zur Folge.
  - **Vorweiblichkeit:** Bei der Vorweiblichkeit ist die Narbe bereits geraume Zeit empfänglich bevor die Staubblätter ihre Staubbeutel entleeren. Dies hat zur Folge, dass die Narbe der Blüte mit einer hohen Wahrscheinlichkeit bestäubt wird noch bevor die Blüte ihre eigenen Pollen abgeben kann.
- **Räumliche Trennung**
  - **Innerhalb einer Blüte:** Die Trennung innerhalb einer Blüte wird dadurch erreicht, dass entweder der Griffel kurz und die Staubblätter lang oder aber der Griffel lang und die Staubblätter kurz sind. Dadurch werden zwei Ebenen in einer Blüte geschaffen, so dass eine Bestäubung nur auf einer Ebene stattfinden kann, d.h. wenn Pollen eines langen Staubblatts auf die Narbe eines langen Griffels oder aber Pollen eines kurzen Staubblatts auf die Narbe eines kurzen Griffels treffen. Somit ist eine Bestäubung innerhalb der gleichen Blüte nicht möglich.
  - **Verteilung auf mehrere Blüten:** Eine weitere Möglichkeit der räumlichen Trennung wird damit erreicht, dass eine Blüte nicht gleichzeitig Pollen und Narbe zur Verfügung stellt, sondern entweder nur Pollen oder nur eine Narbe ausbildet. So kann eine Pflanze einerseits Blüten mit Pollen und andererseits Blüten mit einer Narbe zur Verfügung stellen und dadurch eine Selbstbestäubung innerhalb einer Blüte ausschließen.

    Diese Trennung kann so weit getrieben werden, dass es von einer Pflanzenart (wie z.B. der Kiwi) männliche Pflanzen, die nur Blüten mit Pollen ausbilden, wie auch weibliche Pflanzen, die nur Blüten mit einer Narbe ausbilden, existieren. Diese Pflanzen können sich nur dann vermehren, wenn beide Geschlechter in unmittelbaren, für einen Bestäuber überbrückbaren, Abstand zueinander existieren.

2.3 Modell der künstlichen Blütenbestäubung

Die in [16] beschriebene künstliche Blütenbestäubung, die in dieser Arbeit evaluiert werden soll, leitet sich teilweise von den bisherigen, in Abschnitt 2.2 beschriebenen biologischen Kenntnissen ab und macht an einigen Stellen für das Modell notwendige Erweiterungen. Da ein Modell im

allgemeinen immer die Realität abstrahiert, ist auch in diesem Fall eine Abstraktion der relevanten Komponenten und Abläufe auf ein statisches bzw. dynamisches Modell notwendig.
Das in Abschnitt 2.2 erläuterte biologische Bestäubungsmodell wird in den folgenden Punkten eingeschränkt bzw. vereinfacht:

- Da es wenig sinnvoll erscheint die ganze Welt als ein einziges großes System darzustellen, spiegelt ein künstliches Bestäubungssystem nur eine räumlich beschränkte Bestäubungsumgebung wieder.
- Die Bestäubung wird nur von lebenden Pollenträgern durchgeführt. Wind und Wasser werden weder als Bestäuber noch als beeinflussende Faktoren, z.B. für die Verbreitung der Duftstoffe, in das Modell einbezogen.
- Die Anziehung der Pollenträger basiert nur auf geruchlichen, nicht auf optischen Merkmalen.
- Die Bestäuber können als Belohnung nur Nektar jedoch keine Pollen konsumieren.
- Ein Bestäuber kann jedes beliebige Nektar sammelnde Insekt sein, wobei deren Verhalten immer identisch zu dem Sammelverhalten einer nahrungssuchenden Biene ist.
- Das Modell hat keinen Bienenstock oder Sammelplatz für gesammelte Pollen.

2.3.1 Statisches Modell

Die nach den obigen Einschränkungen verbleibenden relevanten Konzepte sind Pflanzen, Blüten, Pollen, Belohnungen, Reizmittel (hier Duftstoffe) und Bestäuber, welche in Abbildung 2.2 anhand eines Metamodells in Verbindung zueinander gebracht werden.

Das System der Blütenbestäubung

Das System der künstlichen Blütenbestäubung (APS = Artificial Pollination System) ist wie folgt definiert:

$$APS = (G, S, P, O, V)$$

wobei

G = *mögliche Gattungen*
S = *mögliche Arten*
P = *existierende Pflanzen*
O = *Menge der Ordnungen*
V = *existierende Bestäuber*

In der Biologie sind Pflanzen nach verschiedenen, aufeinander aufbauenden, hierarchischen Stufen der Botanik klassifiziert. Darunter befinden sich die Begriffe Gattung, Art, Abteilung, Unterabteilung und viele mehr. Für die künstliche Blütenbestäubung ist eine zu feingranulare Klassifizierung unvorteilhaft, so dass nur Art und Gattung einer Pflanze in das Modell einfließen. Des Weiteren wurde von Linnaeus [5], dem Entwickler der binären Nomeklatur für Pflanzen- und Tierarten, festgehalten, dass in der Pflanzenkunde eine Gattung aus einer oder mehreren Arten und eine Art

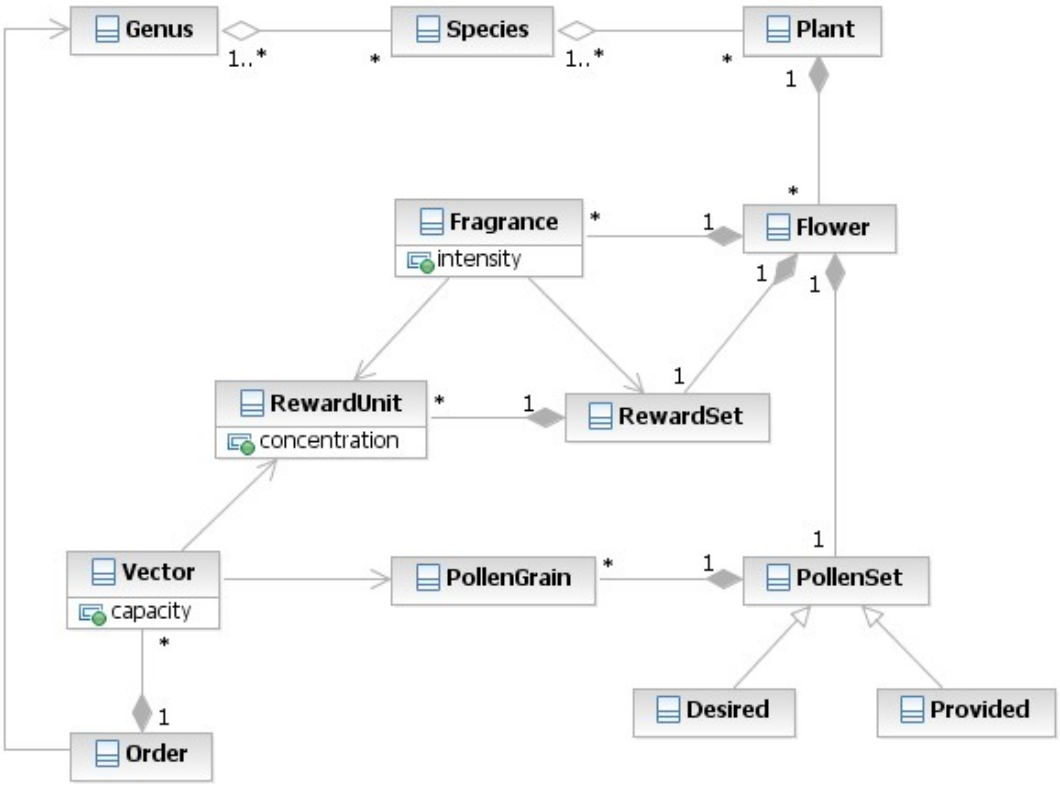

Abbildung 2.2: Das Metamodell eines künstlichen Bestäubungssystems.

wiederum aus einer Vielzahl an Pflanzen besteht. Dabei sind die Mengen der Arten aus denen sich Gattungen zusammensetzen untereinander disjunkt, d.h. eine Art kann in der Natur nicht gleichzeitig zu zwei Gattungen zählen. Für das Modell wurde diese Einschränkung aufgeweicht, so dass eine Gattung aus mehreren Arten bestehen, eine Art aber auch zu mehreren Gattungen zählen kann. Dementsprechend besteht das System der Blütenbestäubung aus einer Menge möglicher Gattungen und deren Arten.

Für die Einordnung im Tierreich hat Linnaeus ebenfalls eine Vielzahl an hierarchischen Stufen zur Klassifizierung aufgestellt, wobei in der künstlichen Blütenbestäubung nur der Begriff der Ordnung eine Rolle spielt. Das Modell besteht demnach aus einer Menge möglicher Ordnungen wie auch deren existierenden Bestäubern, als auch einer Menge an existierenden Pflanzen, die Blüten zur Bestäubung hervorbringen. Da das künstliche Bestäubungssystem eine endliche Bestäubungsumgebung wiederspiegelt, ist die Anzahl an Pollenträgern wie auch Pflanzen immer genau feststellbar.

Die Pflanzen

Eine Pflanze p ∈ P is definiert als

$$p = (PG, PS, F)$$

wobei

PG	$\subseteq$	G
PS	$\subseteq$	S
F	$=$	*Blüten der Pflanze*

Eine Pflanze muss laut Definition aus einer oder mehreren, im Modell vorhandenen Gattungen wie auch Arten bestehen. Des Weiteren kann die Pflanze im Laufe ihrer Existenz mehrere Blüten zur Bestäubung erzeugen, welche im folgenden Abschnitt genauer betrachtet werden.

Die Blüten

Eine Blüte f ∈ F ist definiert als

$$f = (g, s, Pol^{prov}, Pol^{des}, R, A)$$

wobei

g	$\in$	PG
s	$\in$	PS
Pol^{prov}	$=$	*angebotene Pollen*
Pol^{des}	$=$	*nachgefragte Pollen*
R	$=$	*angebotene Belohnungen*
A	$=$	*Menge der Duftstoffe, die die Blüte abgeben kann*

Eine Blüte wird immer von genau einer Pflanze erzeugt und besitzt genau eine Art und Gattung aus der Menge der Arten und Gattungen der erzeugenden Pflanze. Des Weiteren besitzt eine Blüte eine Menge an angebotenen Pollen wie auch eine Menge an nachgefragten Pollen. Für jedes angebotene wie auch nachgefragte Pollenkorn erstellt die Blüte eine Belohnungseinheit, die bei der Entladung oder Abholung an die Biene ausgegeben wird.

Die Pollen

Ein Pollenkorn pol ist definiert als

$$pol = (g, s) \quad (\hat{=}\ pol^{g/s})$$

wobei

g = *Gattung des Pollenkorns*
s = *Art des Pollenkorns*

Ein Pollenkorn enthält genau die Gattung wie auch die Art der Blüte, die es erzeugt hat. Außer diesen beiden Merkmalen enthält es keinerlei Informationen, die Aufschluss auf die Blüte geben, von der das Pollenkorn stammt.

Die Menge an Pollen, die von einer Blüte angeboten wird, ist definiert als

$$Pol^{prov} = \{pol_1^{g/s}, ..., pol_k^{g/s}\}$$

wobei

k = *Anzahl der angebotenen Pollenkörner*

Analog dazu ist die Menge der Pollen, die von einer Blüte für die eigene Bestäubung nachgefragt werden, definiert als

$$Pol^{des} = \{pol_1^{g/s}, ..., pol_l^{g/s}\}$$

wobei

l = *Anzahl der nachgefragten Pollenkörner*

Die Belohnung

Eine Menge R an Belohnungseinheiten ist folgendermaßen definiert

$$R = \{r_1^{g/s/c}, ..., r_{k+l}^{g/s/c}\}$$

wobei für eine Belohnungseinheit r ∈ R gilt

$$r = (g, s, c) \quad (\hat{=}\ r^{g/s/c})$$

mit

g = *Gattung der Blüte, die die Belohnung anbietet*
s = *Art der Blüte, die die Belohnung anbietet*
c = *die Zuckerkonzentration*
$k+l$ = *die Anzahl an angebotenen wie auch nachgefragten Pollen*

Wie bereits angesprochen, werden die Pollenträger in der künstlichen Blütenbestäubung nur durch Nektar belohnt. Eine Belohnungseinheit kann dementsprechend als Nektartröpfchen interpretiert werden, der sich in Bezug auf seine Menge und Zuckerkonzentration unterscheidet. Je süßer der Nektar, d.h. je mehr Energie er gibt, desto bevorzugter wird die Blüte von einem Bestäuber angeflogen. Dasselbe gilt für die Menge des Nektars. In dem künstlichen Modell erhält ein Bestäuber ein Nektartröpfchen für jedes abgegebene bzw. aufgenommene Pollenkorn, d.h. je mehr Pollenkörner ein Bestäuber aufnimmt oder abgibt, desto mehr Nektar erhält er.

Um das Modell zu vereinfachen, wird angenommen, dass die Konzentration der Belohnungseinheiten innerhalb der Menge an angebotenen Belohnungen identisch bleibt. D.h. die Konzentration kann für die komplette Menge an Belohnungen verändert werden, nicht aber für eine einzelne Belohnungseinheit.

Die Duftstoffe

Die Duftstoffe werden von den Blüten abgegeben, um eventuelle Bestäuber anzulocken. Damit ein Bestäuber angelockt werden kann, müssen die relevantesten Informationen der Blüte ihrem Duftstoff mitgegeben werden. Dementsprechend ist ein Duftstoff A folgendermaßen definiert

$$A = (g, s, c, k, l, i)$$

wobei

g	=	*die Gattung der erzeugenden Blüte*
s	=	*die Art der erzeugenden Blüte*
c	=	*die Konzentration der Belohnungen*
k	=	*die Anzahl der angebotenen Pollen*
l	=	*die Anzahl der nachgefragten Pollen*
i	=	*die Intensität des Duftstoffes*

Jeder Duftstoff wird von einer Blüte mit einer bestimmten Intensität ausgesendet, die im Laufe der Zeit linear abnimmt. Durch diese Intensität ist ein Bestäuber in der Lage festzustellen, wie alt die Informationen des Duftstoffes sind. Um dem geeignetsten Duftstoff zu folgen, wertet ein Bestäuber dessen Informationen wie auch Aktualität aus, d.h. er lässt sich von ihm anziehen. Welcher Duftstoff für einen Bestäuber am geeignetsten ist, hängt von der Entscheidungsfunktion und der Gewichtung der einzelnen Informationen ab. Sollte ein Bestäuber bereits einem Duft folgen, so kann er trotzdem seine Richtung ändern sobald er einen geeigneteren Duft ‘schnuppert’.

Die Bestäuber

Ein Bestäuber v ∈ V (im Modell ‘Vector’ genannt) is wie folgt definiert

$$v = (o, cap, VR, VP)$$

wobei

$o \in O$ *die Ordnung, der ein Bestäuber angehört*
$cap =$ *die maximale Menge an Belohnungseinheiten, die ein Bestäuber aufnehmen kann*
$VR =$ *die Menge an aktuell geladenen Belohnungen*
$VP =$ *die Menge an aktuell geladenen Pollen*

Jeder Bestäuber des künstlichen Modells gehört einer Ordnung an. Verglichen mit dem biologischen Modell könnte ein Bestäuber z.B. aus den Ordnungen 'Biene' oder 'Fliege' sein. Durch die Zuordnung spezieller Gattungen VG ist gewährleistet, dass nicht jeder Bestäuber jede Blüte anfliegt, sondern nur bestimmte Blüten bedient werden. Dementsprechend nimmt ein Pollenträger nur Pollen einer Blüte f auf, für deren Gattung g $\in$ VG gilt. Durch die Kapazität wird die mögliche Beladung eines Bestäubers eingeschränkt, so dass er nur eine maximale Anzahl an Belohnungen (bzw. Pollen) transportieren kann. Da ein Bestäuber, wie z.B. eine Biene, in der Natur zu ihrem Stock fliegt sobald sie voll beladen ist, muss dem Bestäuber in dem künstlichen Modell auch ohne Bienenstock die Möglichkeit gegeben werden, seine gesammelten Nektartröpfchen, d.h. Belohnungen, abzugeben. Neben der Entladung hat die Abgabe des Nektars für den Bestäuber die Auswirkung, dass er auf einen neuen Sammelflug gehen kann, ohne an eine spezielle Blütenart gebunden zu sein. Die erfolgreiche Abgabe des Nektars wird in diesem Modell dadurch erreicht, dass ein Bestäuber, sobald er ein Pollenkorn aufnimmt, auch dessen Belohnung aufnimmt. Wenn der Bestäuber das aufgenommene Pollenkorn abgibt, bekommt er erneut eine Belohnung. Sobald der Bestäuber für ein Pollenkorn beide Belohnungseinheiten erhalten hat, konsumiert er diese. Dadurch wird nach erfolgreicher Abgabe eines Pollenkorns Platz für die Aufnahme neuer Pollenkörner frei. Des Weiteren entspricht der Konsum der letzten Belohnungseinheit dem Beginn eines neuen Sammelflugs.

2.3.2 Dynamisches Modell

Wie bereits erwähnt, handelt es sich bei den Pollenträgern des künstlichen Bestäubungsmodells um Insekten, deren Verhalten an das Sammelverhalten einer nahrungssuchenden Biene angenähert ist. Dementsprechend besteht der dynamische Teil des Modells aus der Beschreibung des Verhaltens eines Pollenträgers.

Wie in Abbildung 2.3 zu erkennen ist, kann das Verhalten eines Bestäubers in die beiden Subprozesse 'Nektar aufspüren' und 'Nektar sammeln' aufgeteilt werden.

Nektar aufspüren Da sich ein Bestäuber, der auf der Suche nach Nektar ist, nur anhand von Duftstoffen orientieren kann, nimmt er in einem ersten Schritt alle Duftstoffe in seiner Umgebung auf. Falls er keinen Duft 'riecht', geht er davon aus, dass keine Blüte in der Umgebung ist und bewegt sich zufällig weiter. Sollte er Duftstoffe riechen, so werden diese von ihm nach ihren Informationen, d.h. nach Konzentration, Anzahl der angebotenen bzw. nachgefragten Pollen und Intensität des Duftes evaluiert und dem Besten gefolgt.

Falls der Bestäuber nach diesem Vorgang auf keiner passenden Blüte gelandet ist, so wiederholt er ihn bis er eine für ihn passende Blüte gefunden hat.

Nektar sammeln Befindet sich der Bestäuber nun auf einer passenden Blüte, d.h. einer Blüte, die passende Pollen abgeben will oder nachfragt, so überprüft er im ersten Schritt, ob er für die

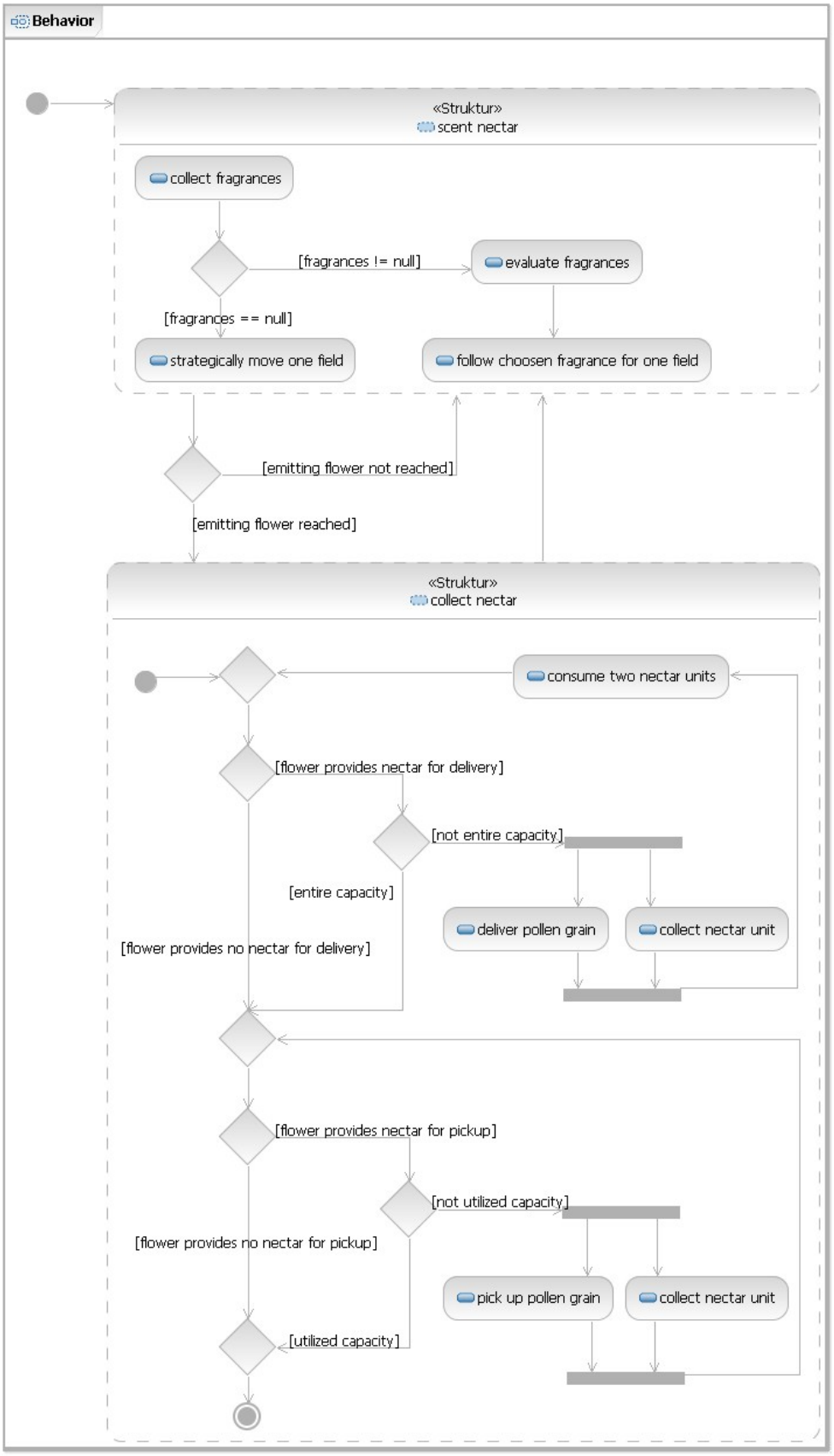

Abbildung 2.3: Das Verhalten eines Bestäubers.

Abgabe seiner Pollen Nektar bekommen kann. Ist dies der Fall, so entlädt er ein Pollenkorn und erhält dafür eine Belohnung. Diese Belohnung wird zusammen mit der Belohnung, die er bei der Aufnahme des abgegebenen Pollenkorns erhalten hat, von dem Bestäuber konsumiert. Der Vorgang des Entladens wird nun so oft durchgeführt bis es keine passenden Pollen zum Entladen mehr gibt.

Ist der Entladevorgang abgeschlossen, überprüft der Bestäuber, ob die Blüte Nektar für die Aufnahme von Pollen gibt. Ist dies der Fall, prüft der Bestäuber, ob er Pollen aufnehmen kann. Ist auch dieser Test positiv, so nimmt der Pollenträger jeweils ein Pollenkorn und eine Belohnung auf. Diesen Vorgang wiederholt er so oft, bis die Blüte keine passenden Pollen mehr zur Verfügung stellt oder seine Kapazität erreicht ist.

Kapitel 3

Simulationsumgebung

Um Erkenntnisse über das formale Modell der künstlichen Blütenbestäubung zu gewinnen, muss es nach speziellen, in Abschnitt 3.1 festgelegten, Kriterien evaluiert werden. Um das formale Modell in diesem Rahmen beurteilen zu können, wird eine passende Simulationsumgebung benötigt, die einerseits die künstliche Blütenbestäubung simulieren, andererseits aber auch relevante Werte für die Evaluation überwachen bzw. speichern kann. Das Anforderungsprofil an die benötigte Simulationsumgebung wird in Abschnitt 3.2 erstellt. Basierend auf diesem Profil werden einige etablierte Simulationsumgebungen bewertet und nach ihrer Einsatzfähigkeit in Bezug auf die künstliche Blütenbestäubung bewertet.

3.1 Evaluationskriterien

Teil dieser Arbeit ist die Evaluierung der künstlichen Blütenbestäubung, deren Aufbau und Ergebnisse in Kapitel 5 allgemein wie auch anhand spezieller Szenarien vorgestellt werden. Dieser Abschnitt dient dazu eine Vorstellung davon zu bekommen, was und wie evaluiert werden soll. Dabei handelt es sich einerseits um eventuelle Eigenschaften des Modells wie auch Überlegungen, anhand welcher Parameter diese Eigenschaften am besten festgestellt werden können.

3.1.1 Skalierbarkeit

> Ein System, das als skalierbar bezeichnet wird, bleibt auch dann effektiv, wenn die Anzahl der Ressourcen und die Anzahl der Benutzer wesentlich steigt. [7]

Die Skalierbarkeit betrachtet den Ressourcenverbrauch eines Systems in Zusammenhang mit steigenden Eingabemengen, bei denen es sich in folgendem Beispiel um die Anzahl der angemeldeten Benutzer handelt. Steigt der Ressourcenverbrauch eines Systems bei jeder weiteren Anmeldung eines Benutzers um einen linearen Faktor, so ist dieses gut skalierbar. Eine exponentielle Steigung des Ressourcenverbrauchs für jeden angemeldeten Benutzer deutet hingegen auf ein schlecht skalierbares System hin.

Da es bei der künstlichen Blütenbestäubung effektiv keine Benutzer gibt, muss die Skalierbarkeit von einem bzw. mehreren anderen Faktoren abhängen. Die variablen und damit für die Skalierbarkeit relevanten Teile des Systems sind u.a. die Bestäuber und Blüten. So sollte auf jeden Fall

gemessen werden, wie sich das System bei einer steigenden Menge an Bestäubern wie auch Blüten verhält. Dabei kann des Weiteren herausgefunden werden, ob und wie, d.h. in welchem Verhältnis, Blüten und Bestäuber am besten miteinander interagieren. Konkret müssen die folgenden Fragen durch Messungen beantwortet werden:

Welche Kosten verursacht ein weiterer Bestäuber bzw. eine weitere Blüte? Hierbei müssen die durchschnittlich verursachten Kosten eines weiteren Bestäubers innerhalb eines festgelegten APS gemessen werden. Dabei muss darauf geachtet werden, wie sich die Auslastung der Bestäuber, die Bestäubungsdauer einer Blüte, die Anzahl der Duftstoffe und die Dauer der Simulation verändert.

Welche Kosten verursacht eine weitere Pflanze? Um die Kosten einer weiteren Pflanze zu berechnen, muss einem festgelegten APS eine weitere Pflanze hinzugefügt werden. In dem neuen APS muss, wie im vorhergehenden Abschnitt, festgestellt werden, wie hoch die Auslastung der Bestäuber, die Bestäubungsdauer einer Blüte, die Anzahl der Duftstoffe und die Dauer der Simulation verändert.

Welcher Zusammenhang besteht zwischen Bestäuber und Blüte? Der Zusammenhang zwischen den Komponenten Bestäuber und Blüte liegt in der Symbiose zueinander, d.h. für eine spezielle Anzahl an Blüten gibt es eine optimale Menge an Bestäubern und umgekehrt. Diese optimale Bestäuber-Blüten-Kombination ist gegeben, wenn das System u.a. die beste Auslastung und die besten Bestäubungszeiten vorweisen kann.

3.1.2 Robustheit

Die Robustheit eines Systems drückt sich dadurch aus, dass es auch unter ungünstigen Bedingungen noch zuverlässig funktioniert. Diese Eigenschaft kann noch gesteigert werden, indem ein System unter ungünstigen Bedingungen nicht nur zuverlässig, sondern auch noch korrekt und zügig funktioniert.

Im Modell der künstlichen Blütenbestäubung gibt es u.a. die folgenden ungünstigen Bedingungen:

- Eine Pflanze verhält sich falsch, d.h. sie produziert z.B. zu viele, zu wenige oder falsche Blüten.
- Eine Blüte verhält sich falsch, d.h. sie produziert z.B. zu viele oder zu wenige Duftstoffe oder Pollen.
- Eine Pflanze bzw. Blüte fällt vollständig aus.
- Ein Bestäuber verhält sich falsch, d.h. er transportiert z.B. zu viele, zu wenige oder falsche Pollen.
- Ein Bestäuber fällt vollständig aus.

Unter der Annahme das bei einem realen Einsatz des Systems die Software fehlerfrei funktioniert kann es dennoch passieren, dass eine ganze Komponente ausfällt. Dies ist unter anderem möglich, wenn die Berechnungen auf Daten eines fehlerbehafteten Sensors basieren, der für die Aufnahme

der Umgebung des Bestäubers bzw. der Duftstoffe der Blüten verantwortlich ist. Wenn eine Komponente nicht den richtigen Input von seinen Sensoren bekommt, wird sie sich dementsprechend falsch verhalten. Ein weiterer Aspekt ist die Umgebung, in der sich ein Bestäuber befindet. Sollte er z.B. als Roboter Dinge aus einem Raum in den nächsten Raum transportieren, so ist es für ihn unerlässlich, dass er in keinem Raum eingesperrt wird. Sollte ein Bestäuber in seiner Umgebung eingesperrt werden, so wird er in diesem Fall seinen Pflichten nicht mehr nachkommen können und fällt dadurch aus. Um die Robustheit des Modells bei einigen der eben beschriebenen Ausfälle festzustellen, müssen u.a. folgende Werte gemessen werden:

1. Der Ausfall einer Komponente entspricht dem Zusammenbruch eines Bestäubers, einer Pflanze oder einer Blüte. In diesem Fall sollte der Durchsatz, d.h. die Anzahl an übertragenen Pollen in einer bestimmten Zeit, des ursprünglichen Systems mit dem des neuen Systems verglichen werden. Aus dieser relativen Veränderung können Schlüsse gezogen werden, in wie fern das System einen Ausfall einer bzw. mehrerer Komponenten toleriert.

2. Der Ausfall eines Sensors für die Aufnahme der Duftstoffe kann unterschiedliche Auswirkungen haben. Sollte der Sensor fehlerhafte Daten liefern, so entspricht dies einem kompletten Ausfall des Bestäubers. Ein Sensor kann beispielsweise den letzten gespeicherten Wert vor seinem Absturz immer wieder ausgeben, ohne dass er neue Werte einliest. Demnach wird dem Bestäuber mitgeteilt, dass er z.B. immer nach links gehen muss um einem Duftstoff zu folgen, obwohl dies nicht der Realität entspricht. Sollte der Sensor jedoch keinerlei Daten mehr liefern, entspricht dies einer zufälligen Suche nach Pollen. Dieses Verhalten kann dadurch erreicht werden, dass der Bestäuber die gerochenen Düfte ignoriert und sich frei ohne Beeinflussung in seiner Umwelt bewegt.

   Die Robustheit des Systems unter Berücksichtigung beider Ausfallarten eines Sensors kann wie in 1 gemessen werden.

3.1.3 Flexibilität

Ein flexibles Softwaresystem zeichnet sich durch seine Portabilität und Anpassungsfähigkeit aus. Im Kontext der künstlichen Blütenbestäubung ist mit Flexibilität jedoch die Anpassungsfähigkeit des Systems selbst und nicht seiner Software gemeint. Eine Anpassung ist notwendig, wenn z.B. eine Komponente des Systems ausfällt. Der Ausfall einer zentralen Komponente kann bei zentraler Verwaltung zum kompletten Absturz des Systems führen. Sollte jedoch eine Komponente in einem verteilten System ausfallen, so sollte dieser Ausfall in einem flexiblen System mit großer Wahrscheinlichkeit abgefangen werden können.

In Zusammenhang mit der künstlichen Blütenbestäubung kann die Flexibilität ähnlich geprüft werden wie die Robustheit in Abschnitt 3.1.2. Mit der Robustheit wird gemessen, ob sich ein System trotz unvorhersehbaren, ungünstigen Bedingungen korrekt verhält. Die Flexibilität hingegen misst die Anpassungsfähigkeit eines Systems bei einer dynamischen Veränderung des Modells. Demnach müssen die Veränderungen des Modells nicht unbedingt negativ, d.h. ein Fehlverhalten oder Ausfall einer Komponente, sein. Wie in der Natur, kann auch in dem Modell der künstlichen Blütenbestäubung beispielsweise eine Blüte von einer Pflanze neu hervorgebracht werden. Nach Entstehung der neuen Blüte kann das System nun auf seine Flexibilität überprüft werden, d.h.

ändert es sein Verhalten auf Grund der neuen Blüte oder ignoriert es die Veränderung. Für eine Evaluation des Modells in Bezug auf seine Flexibilität werden die Veränderungen dynamisch während der Laufzeit künstlich herbeigeführt. Flexibilität kann demnach durch die Reaktion des Systems auf die in Abschnitt 3.1.2 definierten ungünstigen Eigenschaften wie auch auf die folgenden Ereignisse gemessen werden.

- Ein oder mehrere Bestäuber werden dem System hinzugefügt.
- Ein oder mehrere Pflanzen werden dem System hinzugefügt.
- Ein oder mehrere Blüten werden dem System hinzugefügt.

Das System ist flexibel, wenn es sich an eine neue Umgebungen anpassen kann und die gestellten Anforderungen erfüllt.

3.1.4 Effizienz

Ein System ist im Allgemeinen effizient, wenn es möglichst wenig Ressourcen, Zeit und Speicherplatz für die Lösung eines festgelegten Problems benötigt. Für diese Eigenschaften gibt es jedoch bisher keine Vergleichswerte für das System der künstlichen Blütenbestäubung.

Wird hingegen der Ablauf des Modells und nicht das komplette System betrachtet, kann die Effizienz beispielsweise durch das Verhältnis des optimalen Weges eines Bestäubers zu seinem tatsächlich zurückgelegten Weg gemessen werden. Je direkter ein Bestäuber auf eine passende Blüte zu fliegt, desto effizienter ist das System. Des Weiteren kann gemessen werden, wie oft ein Bestäuber ohne Erfolg eine Blüte anfliegt, d.h. wie oft er einem Duftstoff folgt, dessen Informationen nicht mehr mit dem tatsächlichen Zustand der Blüte übereinstimmen.

3.2 Anforderungen

Um das System der künstlichen Blütenbestäubung nach den in Abschnitt 3.1 beschriebenen Kriterien zu evaluieren, wird eine passende Simulationsumgebung benötigt. Durch die Evaluation erhofft man sich Erkenntnisse über die Einsatzfähigkeit des Systems in einem realen Umfeld. Dies setzt voraus, dass die Evaluation in einer an das reale Umfeld angepassten Umgebung, d.h. einer Simulationsumgebung, durchgeführt wird. Diese Simulationsumgebung muss folgende Anforderungen erfüllen:

- Unterstützung der Wechselwirkung zwischen Blüte und Bestäuber. Die Blütenbestäubung besteht im Gegensatz zu anderen Ansätzen aus verschiedenen Arten gleichartiger Individuen und deren Interaktionen untereinander, d.h. die Symbiose zwischen Pflanzen und Bestäubern.
- Hinzufügen bzw. Entfernen der Simulationsobjekte zur Laufzeit. Für eine Evaluation der Flexibilität beispielsweise, muss eine Simulationsumgebung mindestens die Möglichkeit zur Verfügung stellen, Pflanzen, Blüten und Bestäuber zur Laufzeit neu anzulegen bzw. zu löschen

- Veränderung der Umgebung zur Laufzeit. Wie in 3.1.2 erläutert, kann beispielsweise ein Bestäuber in einer realen Umgebung in einem Raum eingesperrt werden. Um diese Aktion ebenfalls simulieren zu können, muss der Benutzer die Möglichkeit haben, die Umgebung dynamisch zu verändern.
- Überwachung der Simulationsobjekte (für eine Evaluation). Für eine Evaluation werden Werte benötigt, die während der Simulation des Systems entstehen. Diese Werte müssen für Auswertungen dem Benutzer entweder in einer Datei, einer Datenbank oder auf eine andere Weise zur Verfügung stehen. Um die Werte während der Simulation festzustellen, muss die Simulationsumgebung einen Überwachungsmechanismus unterstützen.
- Graphische Oberfläche inkl. Anzeige der Simulation. Um dem Benutzer eine hohes Maß an Komfort zu bieten, sollte eine Simulationsumgebung über eine graphische Benutzeroberfläche verfügen, die einem Anwender die Benutzung des Systems erleichtert. Des Weiteren sollte eine sich laufend aktualisierende Anzeige der Simulation vorhanden sein, damit der Benutzer die Möglichkeit hat, die Funktionalität seines Modells zu überprüfen und dessen Wirkungsweise besser zu verstehen.
- Frei zugänglicher Quellcode. Da eine bereits bestehende Simulationsumgebung voraussichtlich nicht perfekt an die Bedürfnisse des Modells der künstlichen Blütenbestäubung angepasst ist, sollte der Quellcode für eventuelle Veränderungen oder Erweiterungen frei verfügbar sein.

Vorausgesetzt der Quellcode der Simulationsumgebung ist frei zugänglich, so kann die graphische Oberfläche wie auch die relevanten Teile für eine Evaluation im Nachhinein dem System hinzugefügt werden.

3.3 Vergleich existierender Simulationsumgebungen

Um eine Simulationsumgebung zu finden, mit der die künstliche Blütenbestäubung evaluiert werden kann, werden verschiedene bereits bestehende Simulationsumgebungen beurteilt. Diese Beurteilung findet auf Basis der in Abschnitt 3.2 beschriebenen Kriterien statt.

3.3.1 AntSim

Mit AntSim kann der Benutzer das Verhalten einer Ameisenkolonie auf Futtersuche simulieren. Das Programm ist aus der Intension heraus entwickelt worden, dem Benutzer ein besseres Verständnis des selbst-organsierenden und emergenten Verhaltens einer Ameisenkolonie durch eine Visualisierung des Vorgangs zu ermöglichen [21]. Neben einer graphischen Darstellung bietet AntSim die Möglichkeit einen oder mehrere Ameisenstaaten in eine Umgebung zu laden und deren (konkurrierendes) Verhalten bei der Nahrungssuche zu beobachten. Auch wenn das System eine Simulation gleichartiger Individuen, in diesem Fall Ameisen, unterstützt, so ist es auf Grund der Einschränkung auf Ameisenstaaten nicht für eine Simulation verschiedenartiger Individuen und damit auch nicht für die Wechselwirkung zwischen Blüte und Bestäuber geeignet. Des Weiteren bietet es dem Nutzer weder die Möglichkeit ein Simulationsobjekt (hier eine Ameise) zur Laufzeit anzulegen oder zu entfernen, noch ist es möglich die einzelnen Objekte zu überwachen. Da der Quellcode des Systems

nicht frei zugänglich ist, wird eine mögliche Erweiterung um notwendige Funktionalität ebenso ausgeschlossen.

3.3.2 SWARM

Getreu seinem Namen, befasst sich die Simulationsumgebung SWARM mit der Simulation von Schwärmen. Dabei repräsentiert ein Schwarm ein komplettes Modell, bestehend aus verschiedenen Agenten verschiedener Agententypen und einer Zeitangabe. Dieser Schwarm kann wiederum als abgeschlossener Agent bzw. als Simulationsobjekt gesehen werden, der Bestandteil eines anderen Schwarms höherer Ebene ist. Dadurch kann ein Modell hierarchisch aufgebaut werden [18]. SWARM bietet einerseits eine graphische Oberfläche wie auch die Möglichkeit einen oder mehrere Schwärme zur Simulationslaufzeit anzulegen und zu löschen. Andererseits ist diese Simulationsumgebung nicht dafür geeignet das komplexe Verhalten der Agenten der künstlichen Blütenbestäubung, d.h. deren Wechselwirkung untereinander, zu simulieren. Konkret handelt es sich dabei nicht um die Bestäuber, sondern um die Pflanzen, die abhängig von ihren Bedürfnissen neue Blüten zur Bestäubung erzeugen. Die Pflanze und die Blüte könnten in SWARM als Agenten umgesetzt werden, deren Beziehung zueinander ist allerdings nicht modellierbar. Nichts desto trotz bietet SWARM ein Observer-System durch das der Benutzer jeden beliebigen Agenten des Modells überwachen kann. Für eventuelle Erweiterungen wird der Quellcode des Systems in Objective-C zur Verfügung gestellt.

3.3.3 SeSAm

SeSAm ist darauf spezialisiert agenten-basierte Simulationen durchzuführen. Dafür werden dem Benutzer eine Modellierungs- wie auch Simulationsumgebung angeboten, die eine einfache Modellierung und anschließende Simulation komplexer Modelle ermöglichen sollen [17]. Durch die graphische Oberfläche ist der Benutzer in der Lage, seine Simulation graphisch zu entwerfen ohne die verwendete Programmiersprache beherrschen zu müssen. SeSAm bietet neben einer sehr ausgereiften graphischen Oberfläche die Möglichkeit sämtliche Simulationsobjekte zu überwachen und deren Verhalten zu analysieren. Des Weiteren ist der Quellcode, geschrieben in Java, frei zugänglich. Jedoch ist SeSAm weder in der Lage, die Wechselwirkung zwischen Blüte und Bestäuber zu simulieren, noch hat der Benutzer die Möglichkeit ein Simulationsobjekt zur Laufzeit zu löschen oder hinzuzufügen.

3.3.4 NetLogo

NetLogo ist eine Simulationsumgebung, die dem Benutzer einerseits eine große Anzahl an vorgefertigten Modellen zur Verfügung stellt, andererseits aber auch die Möglichkeit bietet, ein Modell anhand von ‘turtles’, d.h. gedachte Wesen, die sich auf dem Bildschirm steuern lassen, selbst zu programmieren [25]. Dafür muss sich der Anwender in die verwendete “Pseudoprogrammiersprache”, das turtle-Prinzip wie auch die Schnittstellen zur Umgebung einarbeiten [25]. Neben einer graphischen Oberfläche, über die der Benutzer mit der Simulationsumgebung interagieren kann, bietet NetLogo eine Überwachungsfunktion der einzelnen Komponenten über einen bzw. mehrere Monitore. Jedoch kann die fehlende Möglichkeit ein Simulationsobjekt zur Laufzeit zu löschen oder

hinzuzufügen auch durch Offenlegung des Quellcodes, geschrieben in Java, nicht ausgeglichen werden. Des Weiteren ist NetLogo nicht in der Lage die Wechselwirkung zwischen einem Bestäuber und einer Blüte zu modellieren und zu simulieren.

3.3.5 OMNet++

OMNet++ ist eine diskrete, Ereignis-basierte Simulationsumgebung, die in erster Linie für die Simulation von Kommunikationsnetzwerken entwickelt wurde [11]. Der Quellcode des in C++ geschriebenen Systems ist für nicht kommerzielle Nutzer frei verfügbar. OMNet++ bietet neben der graphischen Darstellung der Simulation einen graphischen Editor, der es dem Benutzer ermöglicht OMNet++-Modelle zu erstellen. Zwar bietet dieses System eine Überwachungsmöglichkeit der einzelnen Simulationsobjekte, jedoch ist es weder dazu geeignet die Symbiose zwischen Bestäuber und Blüte zu simulieren, noch bietet es die Möglichkeit ein Simulationsobjekt zur Laufzeit hinzuzufügen oder zu entfernen.

3.3.6 XRaptor

XRaptor ist eine in C++ geschriebene Simulationsumgebung, die das Verhalten von Agenten in 2- oder 3-dimensionaler kontinuierlicher Umgebung simuliert. Das Verhalten eines Agenten wird über einen konkreten, vom Benutzer programmierten Kontrollkern festgelegt [4]. Demnach bietet das System keinerlei graphische Unterstützung für den Benutzer, da er seine Modelle selbst implementieren muss. Eine Überwachung wie auch das Hinzufügen bzw. Löschen einzelner Simulationsobjekte wird nicht unterstützt. Des Weiteren gibt es keine Möglichkeit die Wechselwirkung zwischen Bestäuber und Blüte zu simulieren. Der Quellcode von XRaptor ist jedoch frei zugänglich.

3.3.7 Repast

Das in Java geschriebene System Repast unterstützt die Simulation von Agenten-basierten Modellen und wird durch einen diskreten Ereignis Scheduler gesteuert. Der Benutzer kann anhand verschiedener Programmiersprachen wie z.B. Java oder C# eigene Modelle zur Simulation entwickeln, in denen die Eigenschaften wie auch das Verhalten der Agenten spezifiziert wird [20]. Repast ermöglicht es dem Benutzer durch eine graphische Oberfläche den Simulationsablauf zu verfolgen und die Eigenschaften wie auch das Verhalten der Agenten zur Laufzeit zu verändern. Jedoch ist es nicht möglich einen Agenten bzw. ein Simulationsobjekt zur Laufzeit komplett zu löschen oder dem Modell hinzuzufügen. Zwar stellt das System die Beobachtung der einzelnen Simulationsobjekte zur Verfügung, jedoch ist es nicht in der Lage die Symbiose der künstlichen Blütenbestäubung zwischen Bestäuber und Blüte zu simulieren. Der Quellcode von Repast steht frei zur Verfügung.

3.3.8 Fazit

Abschließend werden in Tabelle 3.1 die Ergebnisse der vorangegangenen Abschnitte zusammengefasst. Demnach ist SWARM die einzige Simulationsumgebung, die auch ein dynamisches Entfernen und Hinzufügen von Simulationsobjekten zur Laufzeit anbietet. Jedoch ist keine der betrachteten Simulationsumgebungen in der Lage die Wechselwirkung zwischen Bestäuber und Blüte zu unterstützen. Auch die Erweiterung einer bestehenden Simulationsumgebung um die notwendige

Funktionalität ist nicht praktikabel, da der Aufwand voraussichtlich wesentlich größer wäre als die Entwicklung eines neuen Systems. Demnach liegt es nahe eine neue, an das formale Modell der künstlichen Blütenbestäubung angepasste Simulationsumgebung zu entwickeln.

	GUI	Überwachung	löschen hinzufügen	Wechselwirkung	Quellcode
AntSim	+				
SWARM	+	+	+		+
SeSAm	+	+			+
NetLogo	+	+			+
OMNet++	+	+			(+)
XRaptor					+
Repast	+	+			+

Tabelle 3.1: Ergebnisse der geprüften Simulationsumgebungen

Kapitel 4

Simulator für selbst-organisierende Systeme

Für die Entwicklung eines neuen Systems werden in der Regel spezielle Vorgehensmodelle verwendet. Der Entwurf einer neuen Simulationsumgebung wird in dieser Arbeit anhand des Wasserfallmodells [13] durchgeführt. Dieses Modell gliedert die Softwareentwicklung in folgende Phasen: Initialisierung, Analyse, Design, Realisierung, Einsatz und Test. Die in dieser Arbeit genauer beschriebenen Phasen sind die Analyse, das Design und die Umsetzung bzw. Implementierung des neuen Systems. Demnach werden in Abschnitt 4.1 die für die neu zu erstellende Simulationsumgebung relevanten UseCases erläutert, bevor Abschnitt 4.2 die statische, Abschnitt 4.3 hingegen die dynamische Sicht des Systems basierend auf verschiedenen Designentscheidungen wiederspiegelt. Im Anschluss an das Design wird in Abschnitt 4.4 die Realisierung des Systems erläutert.

4.1 Analyse

Eine Simulationsumgebung bietet dem Benutzer die Möglichkeit einen bestimmten Sachverhalt einerseits durch Simulation auf seine Richtigkeit zu prüfen und andererseits durch Anpassung von Parametern zu verändern. Darüber hinaus muss eine Simulationsumgebung nicht nur einen bestimmten Sachverhalt, sondern eine bestimmte Klasse an Sachverhalten simulieren können. Dies bedeutet im Bereich der künstlichen Blütenbestäubung, dass eine passende Simulationsumgebung die Möglichkeit bieten muss, nicht nur ein spezielles APS, sondern jedes beliebige in Abschnitt 2.3 definierte APS zu simulieren. Des Weiteren muss eine für die künstliche Blütenbestäubung passende Simulationsumgebung die folgenden, grundlegenden Anforderungen erfüllen:

Laden der Konfiguration

Eine Konfiguration besteht aus einem APS in Zusammenhang mit seiner Umgebung, d.h. neben den Komponenten des APS (siehe Abschnitt 2.3) werden auch die Standorte spezieller Komponenten, d.h. der Pflanzen und Bestäuber, innerhalb einer Umgebung festgelegt. So ist der Benutzer in der Lage jedes beliebige APS innerhalb seiner Umgebung zu laden.

Starten der Simulation

Um eine Konfiguration zu simulieren, muss dem Benutzer die Möglichkeit gegeben werden die Simulation zu starten.

Stoppen der Simulation

Das Anhalten der Simulation kann aus zwei Gründen erfolgen:

1. Der Benutzer bricht die Simulation manuell ab. Das kann u.a. dann der Fall sein, wenn der Benutzer während der Simulation feststellt, dass er aus Versehen eine falsche Konfiguration geladen hat.

2. Das System hält die Simulation an. Dies kann der Fall sein, wenn alle Blüten innerhalb eines APS bestäubt sind und keine Pollen mehr anbieten oder nachfragen. Unter diesen Umständen würde die Simulation keine weiteren relevanten Zustände erreichen und kann auf Grund dessen von dem System beendet werden.

Der Benutzer muss demnach in der Lage sein, die Simulation manuell abbrechen zu können und die Simulationsumgebung muss ebenfalls in der Lage sein festzustellen, wann die Simulation keine weiteren Ergebnisse mehr liefern kann.

Für eine größere Benutzerfreundlichkeit soll die Simulationsumgebung für die Blütenbestäubung durch eine graphischen Benutzeroberfläche bedienbar sein. Des Weiteren soll der bisherige Ablauf der Simulation auf der Oberfläche dargestellt werden, d.h. es muss dem Benutzer ermöglicht werden die Bestäuber, Pflanzen und deren Verhalten über einen Zeitraum hinweg graphisch verfolgen zu können. Unter der Voraussetzung einer graphischen Benutzeroberfläche mit Darstellung des Ablaufs der Simulation werden die bisher aufgelisteten Anforderungen um die Folgenden ergänzt:

Zurücksetzen der Simulation

Nachdem eine Simulation gestoppt wurde, unabhängig ob manuell oder von der Simulationsumgebung, soll der Benutzer die Möglichkeit haben die gleiche Konfiguration noch einmal zu simulieren. Dadurch kann er an der gleichen Konfiguration mehrere Simulationsläufe durchführen, ohne diese erneut laden zu müssen.

Pausieren der Simulation

Dem Benutzer muss es möglich sein die Simulation und damit auch deren graphische Anzeige zu pausieren. Dadurch kann der Benutzer dem Verhalten des Systems besser folgen.

Fortsetzen der Simulation

Nach einer Pause muss der Benutzer ebenfalls in der Lage sein die Simulation wieder fortzusetzen.

Wiederholen der Simulation

Nach einer erfolgreichen Simulation ist es von Vorteil, wenn sich der Benutzer den Ablauf noch einmal abspielen lassen kann, da er während der Simulation eventuell relevante Abläufe nicht wahrnahm oder ihnen nicht folgen konnte.

Verlangsamen / Beschleunigen der Simulation

Durch das Verlangsamen bzw. Beschleunigen der Anzeige ist dem Benutzer die Möglichkeit gegeben, Einfluss auf die Schnelligkeit der Simulation zu nehmen. Dies bietet ihm eine weitere Möglichkeit dem Verhalten des Systems besser folgen zu können.

Visualisierung der Simulation

Da der Ablauf des Modells ohne Visualisierung der Simulation schlechter verständlich ist, wird eine Anzeige grundsätzlich empfohlen. Die Darstellung jedes Simulationsschritts verlangsamt jedoch die Simulation, so dass der Benutzer, dessen Hauptaugenmerk auf der Simulation und nicht dem Verständnis liegt, in der Lage sein muss die visuelle Darstellung zu deaktivieren.

Erhöhen / Verringern der Schrittgröße

Ein möglicher Kompromiss zwischen der graphischen Anzeige jedes Simulationsschritts und keiner graphischen Anzeige bildet das Erhöhen bzw. Verringern der Schrittgröße. Durch eine Erhöhung der Schrittgröße wird die Anzeige seltener aktualisiert und spiegelt nicht jeden einzelnen Schritt wieder. Dies hat zum Vorteil, dass die Simulation weniger Zeit benötigt, dem Benutzer aber dennoch die Möglichkeit geboten wird, der Simulation zu folgen.

Vergrößern / Verkleinern der Anzeige

Bei einer großen Menge an Simulationsobjekten, hier Pflanzen, Blüten und Bestäuber, in einer großen Umgebung, kann die Darstellung der kompletten Umgebung unvorteilhaft sein, da die Objekte zu klein geraten. Für diesen Fall muss dem Benutzer die Möglichkeit gegeben werden, die Anzeige zu vergrößern, d.h. nur eine kleinere Umgebung vergrößert zu sehen.

Anpassung der Parameter

Eine Simulationsumgebung ist nicht nur für die Simulation eines Sachverhalts zuständig, sondern muss auch die Möglichkeit bieten, relevante Parameter einer Simulation zu verändern. Ein Parameter kann beispielsweise die Anzahl der Bestäuber oder die Intensität der ausgesandten Duftstoffe repräsentieren. Diese Parameter sollen offline, d.h. vor Beginn der Simulation, oder online, d.h. während der Simulation, verändert werden können.

Bearbeiten einer Konfiguration

Da das Modell der künstlichen Blütenbestäubung viele Zusammenhänge besitzt, die bei der Erstellung einer neuen Konfiguration leicht übersehen werden können, sollte eine Simulationsumgebung

dem Benutzer die Möglichkeit bieten neue Konfigurationen zu erstellen und bereits bestehende zu verändern. Dabei sollte eine Überprüfung der Konformität der Konfiguration und des Modells automatisch durchgeführt werden. So ist der Benutzer auch ohne genaues Wissen der Zusammenhänge in der Lage eine zu dem Modell konforme und simulierbare Konfiguration zu erzeugen.

Speichern einer Konfiguration

Nachdem eine Konfiguration neu erzeugt bzw. abgeändert wurde, muss es die Simulationsumgebung ermöglichen die neue Konfiguration abzuspeichern.

Überwachen der Simulationsobjekte

Um das Verständnis des Benutzers für das Verhalten eines Modells zu erhöhen, soll es möglich sein die Simulationsobjekte, hier Pflanze, Blüte und Bestäuber, zu überwachen, d.h. Informationen über das Objekt anzeigen zu lassen.

Hinzufügen / Entfernen der Simulationsobjekte

Wie in Abschnitt 3.1 beschrieben wurde, kann die Flexibilität eines Systems getestet werden, indem zur Laufzeit eine Komponente entfernt oder hinzugefügt wird. Dies entspricht teilweise der "Anpassung der Parameter", wobei hier die Auswahl eines speziellen Objekts, das z.B. entfernt werden soll, berücksichtigt wird.

4.2 Architektur

Die Architektur eines Systems spiegelt den Aufbau bzw. die innere Struktur des Systems wieder. Dabei wird jedoch kein Wert auf dynamische Aspekte gelegt und es werden nur Klassen wie auch deren Verbindungen untereinander betrachtet. Die Architektur eines Systems entwickelt sich meist im Laufe der Designphase, da der Entwickler bei der Umsetzung spezieller Anforderungen einige Designentscheidungen treffen muss, die für die Architektur des Systems relevant sind. Auch wenn bisher keine Designentscheidungen getroffen wurden, soll in diesem Abschnitt auf die Architektur vorgegriffen werden, damit Entscheidungen im Design in den folgenden Abschnitten besser nachvollziehbar sind.
Das in Abbildung 4.1 dargestellte Diagramm zeigt den Aufbau des Simulator für selbstorganisierende Systeme. Die für das System wichtigste Komponente ist der Systemkern (siehe *system*). Dieser wird durch eine austauschbare graphische Benutzeroberfläche auch für ungeübte Benutzer leicht bedienbar. Die Kommunikation zwischen beiden Komponenten basiert auf dem Model-View-Controller Pattern, welches im Zusammenhang mit anderen eingesetzten Designmustern in Abschnitt 4.2.5 erklärt wird.

4.2.1 Systemkern

Innerhalb des Systemkerns befindet sich das abstrakte Modell der künstlichen Blütenbestäubung (siehe *system.model.pollination*) wie auch dessen konkrete Szenarien. Wie in Abbildung 4.1 zu sehen

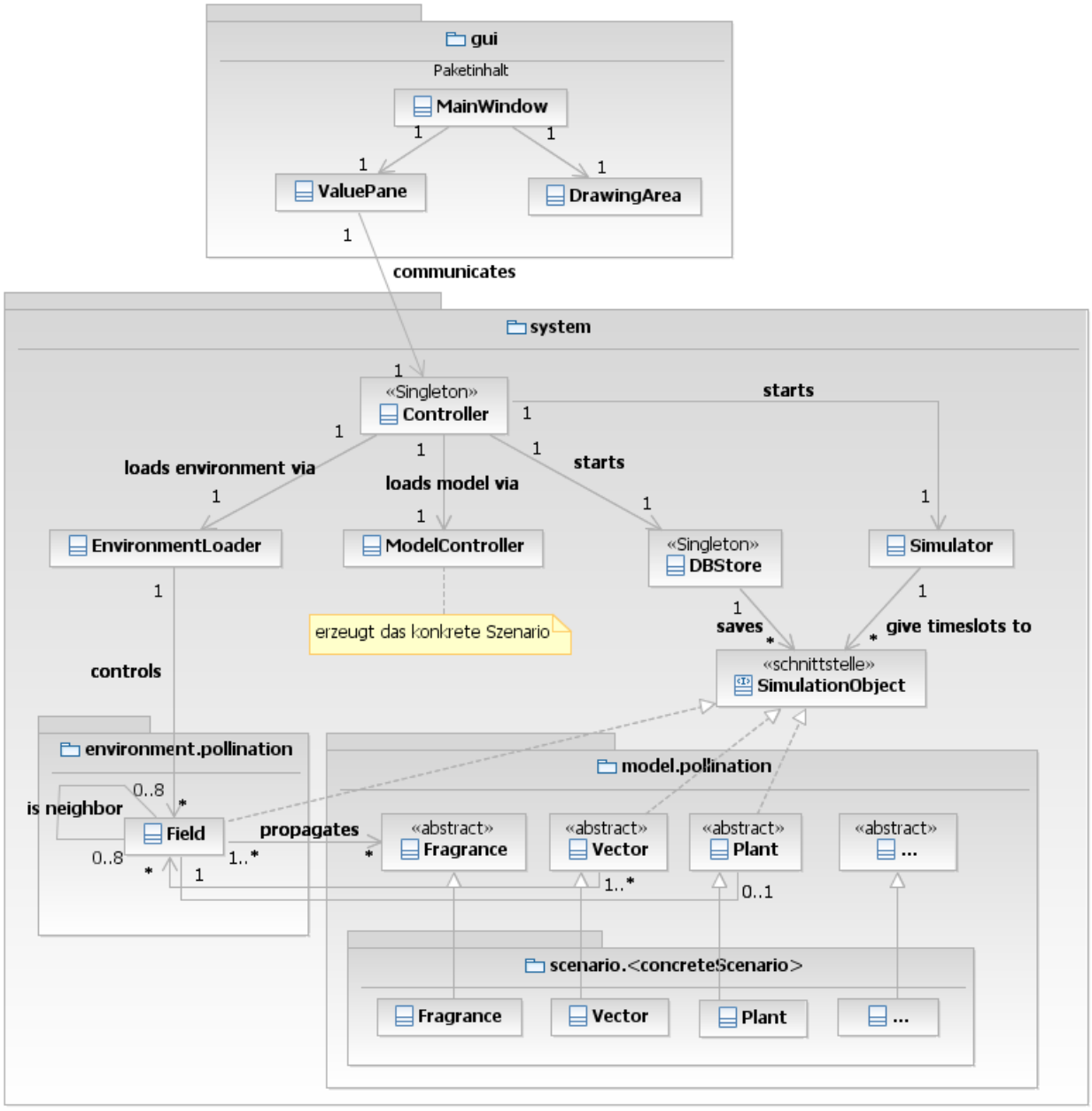

Abbildung 4.1: Architektur des Simulator für selbst-organisierende Systeme.

ist, ist weder das abstrakte Modell noch eines der konkreten Szenarien vollständig dargestellt. Dies beruht darauf, dass noch kein konkretes Szenario existiert, da es erst in Kapitel 5 im Zusammenhang mit der Evaluation benötigt und erstellt wird. Das abstrakte Modell hingegen hätte die Darstellung der Architektur noch unübersichtlicher gemacht, weshalb es in einem eigenen Abschnitt beschrieben wird.

Der Systemkern besteht u.a. aus einem *Controller*, der die Anfragen der graphischen Benutzeroberfläche (siehe Abschnitt 4.2.4) entgegen nimmt und als Singleton (siehe Abschnitt 4.2.5) umgesetzt ist. Für das Anlegen bzw. Laden eines Modells benötigt der *Controller* die Klasse *ModelController*, die alle notwendigen Informationen der konkreten Szenarien besitzt. Damit ein Bestäuber eine Pflanze bzw. Blüte finden kann, benötigt er eine gewisse Orientierung, d.h. er muss sich in einer festgelegten Umgebung bewegen können. Diese Umgebung wird durch den *EnvironmentLoader* neu erstellt oder aus einer Datei geladen. Der genaue Aufbau der Umgebung findet sich in Abschnitt 4.2.2 wieder.

Für die Simulation wird die Klasse *Simulator* verwendet, die nebenläufig in einem eigenen Thread über die *SimulationsObjekte* iteriert. Ein *SimulationsObjekt* kann eine *Pflanze*, ein *Bestäuber* oder ein *Feld* aus der Umgebung sein. Diese Klassen sind neben den *Blüten* und *Duftstoffen* die einzigen Komponenten, deren Zustand sich während einer Iteration verändert. Die *Blüten* werden über ihre *Pflanzen*, die *Duftstoffe* über die Umgebung verwaltet.

Für die Evaluation des Modells existiert die Klasse *DBStore*, die regelmäßig unter Verwendung des Observer-Patterns (siehe 4.2.5) von dem *Simulator* über den Abschluss einer Iteration benachrichtigt wird. Nach jeder Benachrichtigung werden die für die Evaluation relevanten Daten der *SimulationsObjekte* gespeichert, um am Ende einer Simulation eine Bewertung durchführen zu können. Die Klasse *DBStore* ist ebenfalls als Singelton (siehe 4.2.5) umgesetzt, da bei einer eventuellen Anbindung an eine Datenbank nur eine Klasse für das Schreiben zuständig sein sollte.

4.2.2 Umgebung

Die Umgebung einer Simulation entspricht einem großen Gridfeld, welches aus mehreren kleinen *Feldern* besteht. Diese *Felder* können untereinander verbunden sein und markieren dadurch die möglichen Wege eines *Bestäubers* oder *Duftstoffes* innerhalb der Umgebung. Jedes *Feld* kann maximal acht Nachbarn besitzen. Die Ränder des Grids sind abgeschlossen, d.h. man kann z.B. nicht auf der linken Seite aus dem Grid laufen und rechts wieder hineinkommen. Da die *Duftstoffe* sich ebenfalls über die *Felder* der Umgebung ausbreiten, müssen die beiden Komponenten durch eine Assoziation verbunden sein. Da ein *Duftstoff* jedoch als Nachricht ohne eigene Funktionalität angesehen wird, obliegt es dem *Feld* die *Duftstoffe* zu halten, deren Intensität zu verringern und sie weiter zu propagieren. Des Weiteren müssen sich auch *Pflanzen* und *Bestäuber* in der Umgebung orientieren bzw. darin existieren können. Dafür werden ebenfalls Assoziationen zwischen einem *Feld* und den beiden Komponenten eingefügt. Da die Umgebung möglichst wenig Informationen über das Modell haben und dessen Beeinflussung auch nicht zu stark sein sollte, kennen *Bestäuber* wie auch *Pflanzen* das *Feld*, auf dem sie sich befinden, aber das *Feld* nicht die beiden Modellkomponenten. Einzige Ausnahme ist die Verbindung zwischen *Feld* und *Pflanze*, die für eine effizientere Darstellung der Oberfläche auch die Rückrichtung zulässt.

4.2.3 Abstraktes Modell

Das formale Modell der künstlichen Blütenbestäubung wurde bereits in Abbildung 2.2 in Abschnitt 2.3 dargestellt. Auf Basis dieses Modells wird das abstrakte Designmodell der künstlichen Blütenbestäubung entwickelt.
Wie in Abbildung 4.2 zu sehen ist, fehlen in dem Designmodell die beiden Klassen *RewardSet* und *PollenSet*, da beide "nur" eine Menge an Objekten einer anderen Klasse dargestellt haben. Eine Menge an Objekten kann auch durch vorgefertigte Datenstrukturen, wie z.B. ein Array oder einen Vector, umgesetzt werden, so dass die beiden Klassen nicht notwendig sind. So kennt jede Blüte nun direkt ihre angebotenen *Belohnungseinheiten* wie auch ihre angebotenen bzw. benötigten *Pollenkörner*.
Des Weiteren wurde die Klasse *Duftstoff* verändert. Da eine *Blüte* einen *Duftstoff* nur erzeugt, aber ihn sich nicht merkt, wurde die Verbindung zwischen den beiden Klassen entfernt. Außerdem besitzt ein *Duftstoff* keine *Pollenkörner* oder *Belohnungseinheiten*, sondern nur deren Anzahl, was wiederum die Assoziationen zwischen den Klassen hinfällig werden lässt. Der *Duftstoff* besitzt demnach nur noch Assoziationen zu seiner Art und Gattung und hält sämtliche weiteren relevanten Informationen als Attribute.
Eine weitere Veränderung ist die Umstrukturierung der Aggregationen bzw. Kompositionen. Eine Aggregation sagt aus, dass ein Objekt Teil eines anderen großen Objekts ist, aber im Gegensatz zur Komposition auch allein existieren kann. Demnach sind die *Pollenkörner* wie auch *Belohnungseinheiten* Teile einer *Pflanze*, können jedoch auch ohne sie existieren. Eine *Blüte* hingegen ist ein Teil einer *Pflanze* und wird ohne diese nie allein existieren können.
Auch die Aggregationen zwischen den Komponenten *Gattung*, *Art* und *Pflanze* sind weggefallen. Dies beruht darauf, dass auch wenn sich eine *Art* in der Biologie über ihre *Pflanzen* wie auch ihre *Gattung* identifiziert, dies nicht für das künstliche Modell der Blütenbestäubung relevant ist. Die *Art* muss demnach nicht wissen, welche *Pflanzen* zu ihr gehören. Vielmehr ist es für die *Pflanzen* von Relevanz zu welcher *Art* sie gehören. Das Gleiche gilt für die *Gattung*. Eine *Gattung* muss nicht wissen, welche *Arten* zu ihr gehören, aber eine *Art* sollte sich darüber klar sein, welchen *Gattungen* sie angehört.

4.2.4 Graphische Benutzeroberfläche

Die graphische Benutzeroberfläche setzt sich grundlegend aus den drei Klassen *MainWindow*, *ValuePane* und *DrawingArea* zusammen. Das *MainWindow* beinhaltet die beiden Komponenten *ValuePane* und *DrawingArea* und entspricht dem Hauptfenster, welches angezeigt wird. Die Komponente *ValuePane* bietet dem Benutzer eine Übersicht wie auch Änderungsmöglichkeiten der Modellparameter. Des Weiteren wird über diese Komponente der Ablauf der Simulation gesteuert, d.h. der Benutzer ist in der Lage die Simulation u.a. zu starten und zu stoppen. Die Komponente *DrawingArea* bietet eine Anzeige des Ablaufs der Simulation. Dafür wird sie regelmäßig von dem Simulator über das Observer-Pattern (siehe 4.2.5) benachrichtigt.

Für die Darstellung der einzelnen *SimulationsObjekte*, muss die Komponente Informationen über diese besitzen. Wie in Abbildung 4.3 zu sehen ist, werden die notwendigen Informationen der *SimulationsObjekte* durch Interfaces der Benutzeroberfläche zur Verfügung gestellt. So hat die Komponente *DrawingArea* die Möglichkeit alle für sie relevanten Informationen nachzufragen, kann

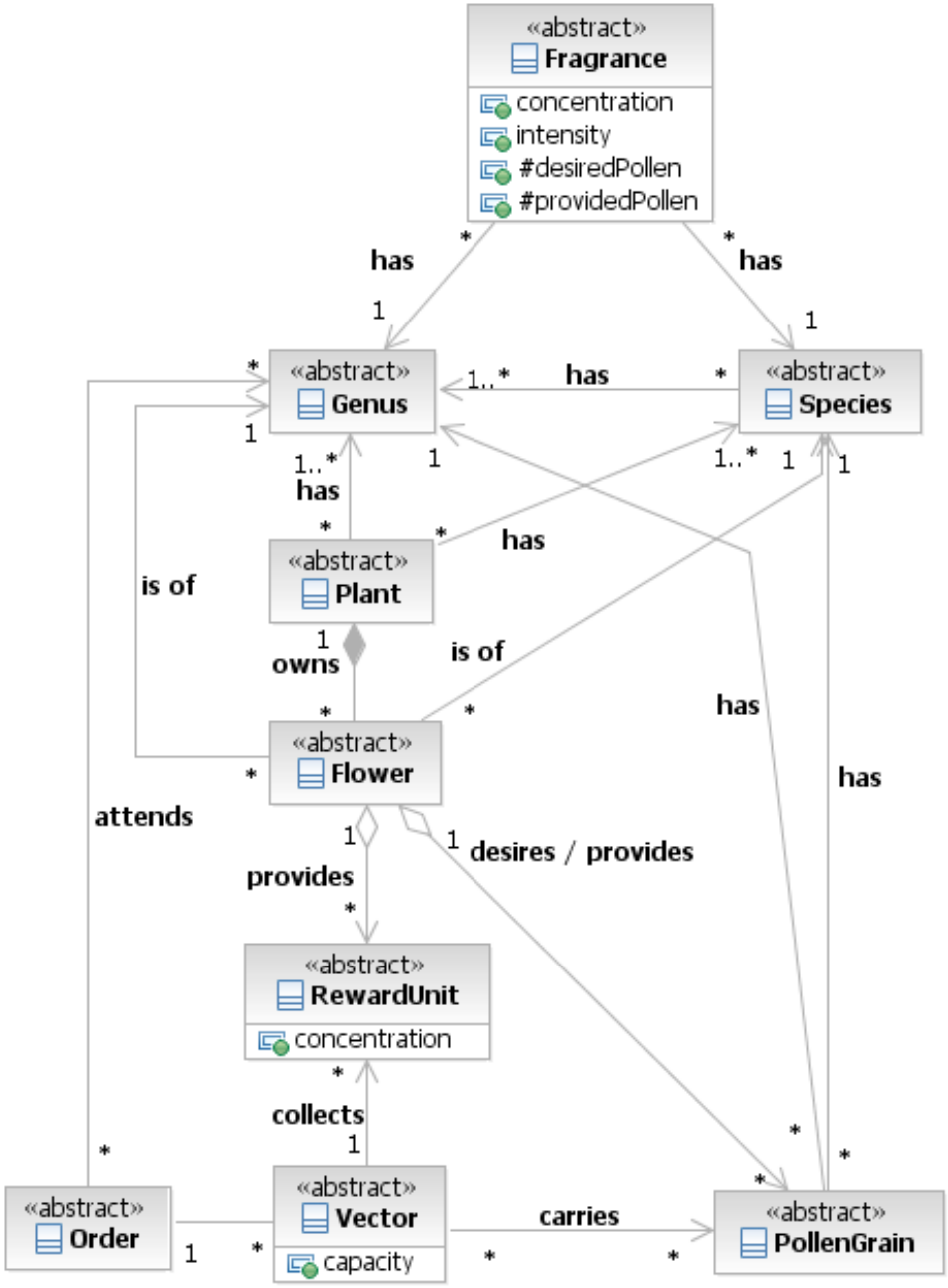

Abbildung 4.2: Aufbau des Bestäubungsmodells.

diese aber nicht ändern.

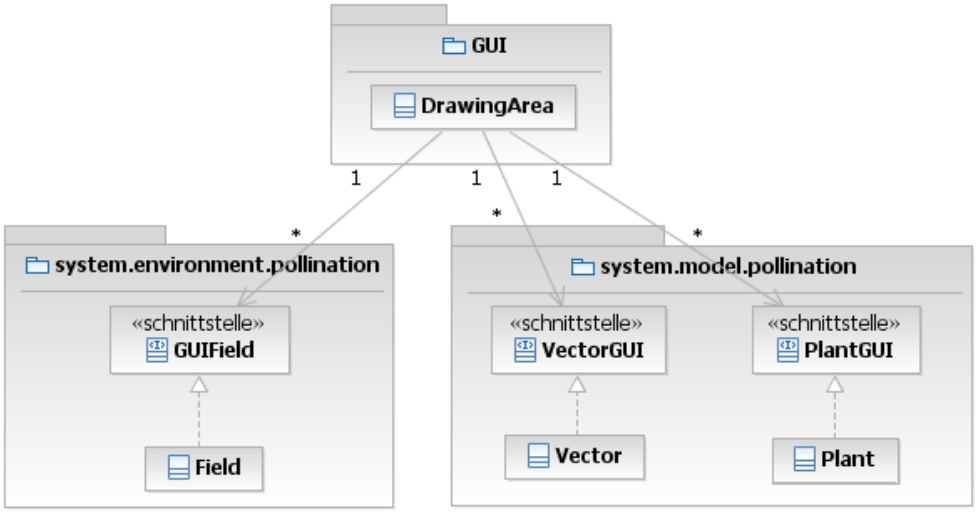

Abbildung 4.3: Zusammenhang zwischen einzelnen Systemkomponenten und der graphischen Benutzeroberfläche.

4.2.5 Eingesetzte Designmuster

Um die Architektur eines Systems besser zu strukturieren, gibt es so genannte Designmuster bzw. Patterns. Diese Designmuster bieten allgemeine Vorschläge, wie ein System unter bestimmten Voraussetzungen möglichst gut strukturiert aufgebaut sein sollte [12]. In diesem Abschnitt werden demnach die in der Architektur der neuen Simulationsumgebung eingesetzten Patterns beschrieben.

Model-View-Controller

Das Model-View-Controller Pattern ist eher ein Architekturparadigma als ein Pattern. Es basiert auf der Trennung des Systems in drei verschiedene Einheiten: das Modell, die Ansicht und die Steuerung. Das Modell ist dabei für die Datenhaltung zuständig, d.h. es besitzt alle relevanten Daten des Systems. Die Ansicht entspricht der graphischen Benutzeroberfläche, die die Daten des Modells gefiltert dem Benutzer repräsentiert. Die Repräsentation des Modells muss jedoch zu jedem Zeitpunkt konsistent zu dem Datenmodell sein, d.h. eventuelle Veränderungen der Daten müssen von der Darstellungsschicht festgestellt werden. Die dritte Einheit ist die Steuerung des Systems. Dies macht meist ein einzelner Controller, der dafür verantwortlich ist einerseits die Daten der Oberfläche zur Verfügung zu stellen, andererseits auch Änderungen des Benutzers an das Modell weiter zu geben. Über den Controller kann demnach auf das Modell in eingeschränktem Rahmen zugegriffen werden. Sollte ein Benutzer über die Oberfläche die Daten ändern, so leitet die Oberfläche die Anfrage an den Controller weiter, welcher die Datenänderung am Modell vornimmt. Auf Grund der Änderung informiert das Modell über das Observer-Pattern die Oberfläche, dass es geändert

wurde. So kann die Oberfläche die angezeigten Daten erneut vom Controller anfordern und in der Darstellung aktualisieren.

Singleton

Ein Klasse, die das Singleton Pattern umsetzt, stellt sicher, dass es nur eine einzige Instanz von ihr gibt. Für den Zugriff auf diese einzelne Instanz bietet die Klasse einen globalen Zugriffspunkt, der durch eine globale Funktion oder statische Methode implementiert sein kann. In der Architektur der neuen Simulationsumgebung sind die Klassen *Controller* und *DBStore* als Singletons umgesetzt. Der *Controller* stellt für die graphische Benutzeroberfläche eine Schnittstelle zum Systemkern zur Verfügung. Es sollten alle Anfragen an den Systemkern über eine Instanz des *Controller* laufen, so dass dieser als Singleton umgesetzt sein muss. Die Klasse *DBStore* hingegen ist für die Speicherung relevanter Daten in eine Datenbank zuständig. Um keine parallelen Zugriffe auf die Datenbank zu gewähren, ist nur eine Instanz der Klasse *DBStore* für das korrekte Abspeichern zuständig.

Observer

Durch das Observer Pattern bzw. Subscribe/Notification Pattern ist eine Klasse dazu in der Lage eine andere Klasse zu "überwachen". Dabei meldet sich die überwachende Klasse (Observer) bei der zu überwachenden Klasse (Observable) an. Sobald die zu überwachende Klasse ihren Zustand geändert hat, meldet sie dies ihren Überwachern, die auf die Meldung eventuell reagieren.

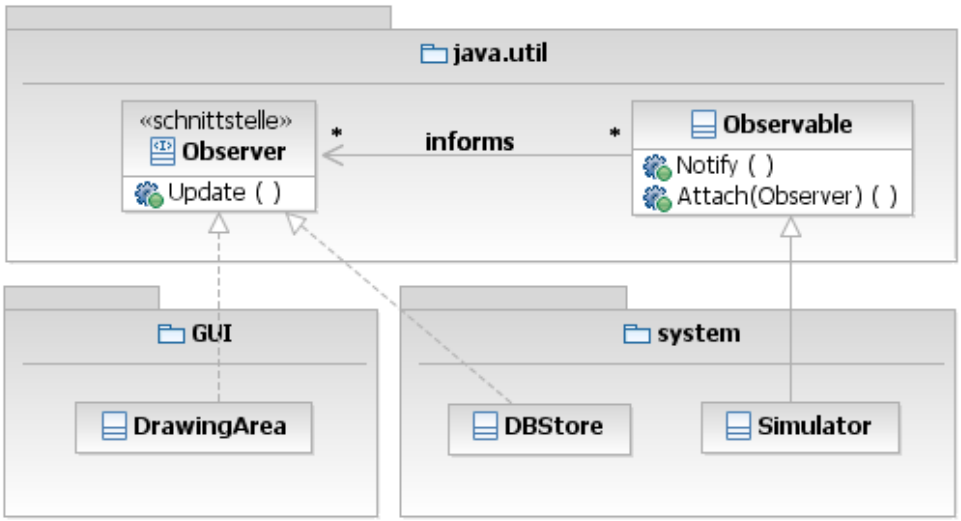

Abbildung 4.4: Verwendung des Observer-Patterns innerhalb der Architektur der neuen Simulationsumgebung.

Das in Abbildung 4.4 dargestellte Modell zeigt das Observer Pattern innerhalb der Architektur der neuen Simulationsumgebung. Die Klassen *DBStore* und *DrawingArea* melden sich bei dem *Simulator* an. Dieser benachrichtigt die beiden Überwacher am Ende jeder Iteration, so dass die Komponente *DrawingArea* ihre Anzeige auf den aktuellen Stand bringen und die Klasse *DBStore* die neuen Zustände der *SimulationsObjekte* abspeichern kann.

4.3 Ergänzende Designentscheidungen

Das Design eines Systems beschäftigt sich neben dem strukturellen Aufbau auch mit dem dynamischen Ablauf. Dabei erhält jede Komponente durch Identifizierung relevanter Methoden ihre Funktionalität.
In dem in Abbildung 4.5 dargestellten Designklassendiagramm wurde Wert darauf gelegt, dass möglichst alle relevanten Methoden der einzelnen Komponenten des Systems auftauchen. Aus diesem Grund sind, der Übersichtlichkeit halber, die Namen und Multiplizitäten der Assoziationen weggefallen. Gemäß dem Aufbau des vorangegangenen Abschnitts 4.2 wurde auf eine detaillierte Darstellung des Bestäubungsmodells innerhalb des allgemeinen Designklassendiagramms verzichtet, so dass dieses in einem eigenen Abschnitt genauer besprochen werden kann.

Bei der dargestellten Funktionalität der einzelnen Komponenten des Systems muss unterschieden werden, ob die Komponente und ihre Funktionalität zu der eigentlichen Simulation oder aber zu der Simulationsumgebung gehören. So ist die Funktionalität fast aller Komponenten im Paket *system* für die Simulationsumgebung relevant, die Funktionalität der Klassen aus den Paketen *system.model.pollination* wie auch *system.environment.pollination* ist jedoch hauptsächlich während der Simulation notwendig.

4.3.1 Relevante Funktionalität der Simulationsumgebung

Wie in Abbildung 4.5 zu sehen ist, bietet der Controller, d.h. die einzige Schnittstelle zu dem System, u.a. die Möglichkeit eine Modell zu laden, eine Konfiguration zu erzeugen, den Simulator zu starten und bei Bedarf diesen wieder zu stoppen.
Für einen reibungslosen Aufbau eines neuen Simulationsexperiments muss der Controller in einem ersten Schritt das Modell laden. Dafür wird eine Instanz des ModelController erzeugt, der das momentan aktuelle Modell wie auch Szenario übergeben bekommt. In einem zweiten Schritt muss nun eine Konfiguration erzeugt werden. Bei einer Konfiguration handelt es sich um eine konkrete Instanz eines Szenarios in einer festgelegten Umgebung. Für eine Konfiguration muss demnach in einem ersten Schritt eine Umgebung geladen werden. Die Erstellung einer neuen Umgebung wird von der Klasse EnvironementLoader übernommen. Diese kann anhand eines Dateipfads eine Umgebung laden. Falls kein Pfad angegeben sein sollte, erzeugt die Klasse ein vollständig verbundenes 10x10 großes Gridfeld. Nach erfolgreichem Laden oder Erstellen einer neuen Umgebung wird diese an den Controller zurück gegeben, da dieser für das Modell eine Zugriffsmöglichkeit auf die Umgebung benötigt. Um eine Konfiguration vollständig zu erzeugen, lässt der Controller erst ein Modell und danach die Konfiguration von der Klasse ModelController erstellen. Für die Erstellung eines Modells bzw. die Instantiierung eines Szenarios eines Modells wird dem ModelController die Art der Konfiguration mitgegeben. Durch diesen Parameter wie auch das Modell und Szenario, welche er bei seiner Erzeugung übergeben bekam, ist der ModelController in der Lage eine konkrete Instanz zu erzeugen. Genaueres zu möglichen Instanzen eines Szenarios findet sich in Kapitel 5. Nachdem nun eine Instanz erzeugt wurde, muss sie in die bereits geladene oder neu erstellte Umgebung eingebettet werden. Dies wird ebenfalls von der Klasse ModelController durchgeführt, wobei hier ein Zugriff auf die Umgebung notwendig ist. Beispielsweise kann in diesem Zusammenhang die Pflanze p auf das Feld f mit den Koordinaten [1,3] gesetzt werden. Für den Modelcontroller bedeutet dies, dass er Informationen über die Umgebung haben muss, was ihm durch den Zugriff

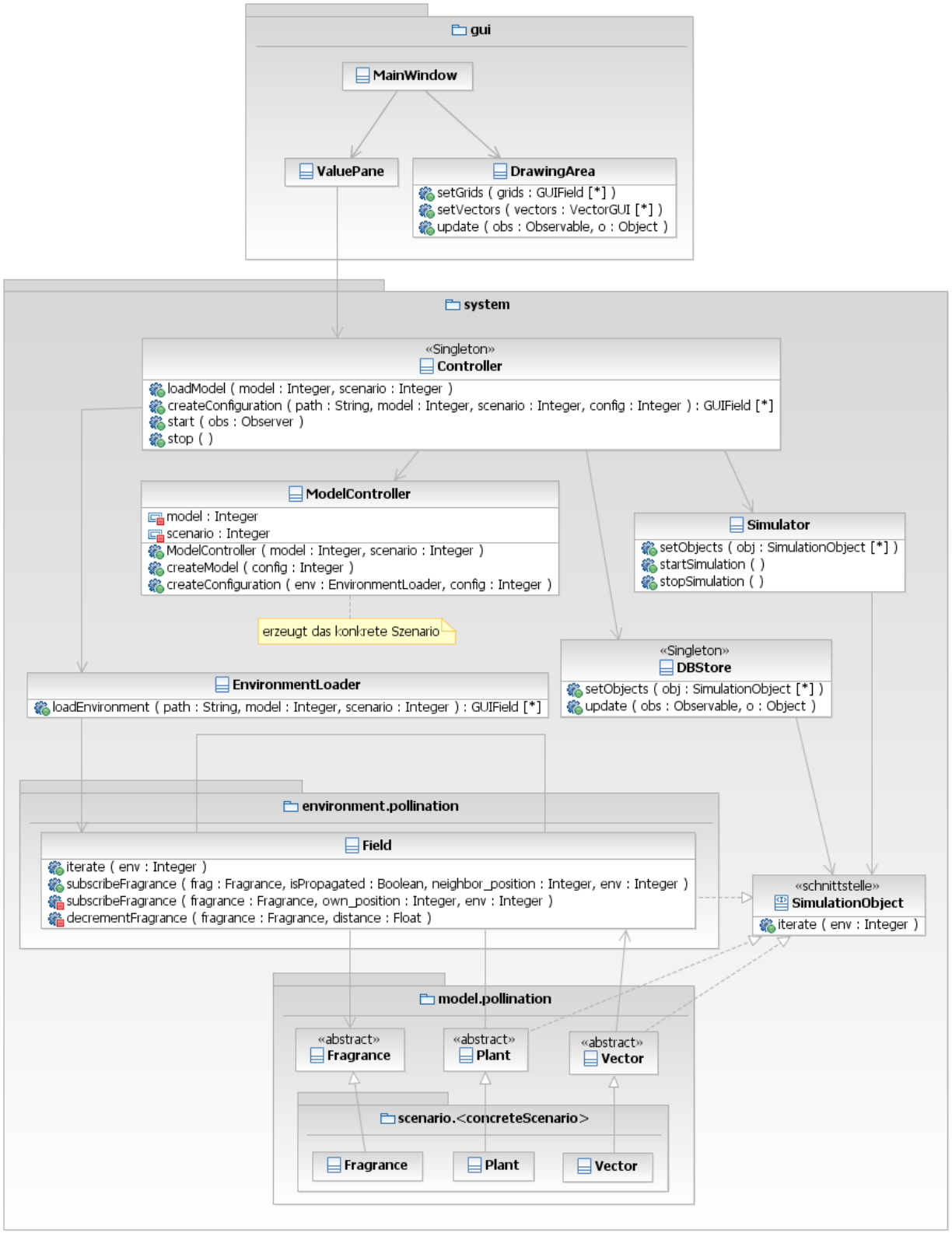

Abbildung 4.5: Designklassendiagramm der neuen Simulationsumgebung.

auf die Umgebung gewährleistet wird. Des Weiteren muss er der neu erzeugten Pflanze mitteilen auf welchem Feld sie steht, und dem Feld eine Referenz der Pflanze zur Verfügung stellen, die auf ihm wächst.

Nach dem erfolgreichen Laden von Modell und Szenario und der Erstellung einer neuen Konfiguration ist nun das Simulationsexperiment so weit vorbereitet, dass der Simulator beginnen kann. Dafür muss dem Controller durch den Aufruf der Methode *start(Observer obs)* mitgeteilt werden, dass keine weiteren Änderungen an dem Simulationsexperiment durchgeführt werden und die Simulation starten kann. Der Controller erzeugt demnach einen neuen Simulator, übergibt ihm die in dem Modell vorhandenen SimulationsObjekte, d.h. alle Pflanzen, Bestäuber und Felder, und meldet einen Überwacher (Observer) bei ihm an. Der Überwacher wird durch das im Simulator implementierte Observer-Pattern regelmäßig benachrichtigt, wenn eine Iteration beendet wurde. Nach dieser Vorarbeit kann der Simulator nun durch *startSimulation()* gestartet werden.

4.3.2 Relevante Funktionalität der Simulation

Während der Simulation lässt der Simulator jedes SimulationsObjekt durch die Methode *iterate(int env)* einen Iterationsschritt durchführen. Der Parameter *env* wird erst während der Implementierung für die Simulation relevant, und aus diesem Grund erst in Abschnitt 4.4 behandelt. Welche Funktionalität die einzelnen SimulationsObjekte bzw. die Klassen des Bestäubungsmodells besitzen müssen, wird ergänzend zu Abbildung 4.5 in Abbildung 4.6 dargestellt.
Um nachvollziehen zu können, welche Schritte während einer Simulation stattfinden, werden die SimulationsObjekte nacheinander genau betrachtet.

Pflanze

Während eines Iterationsschrittes überprüft eine Pflanze ihre Blüten. Sollte eine Blüte keine Pollenkörner mehr anbieten bzw. nachfragen, wird diese Blüte gelöscht. In einem zweiten Schritt lässt die Pflanze ihre Blüten ebenfalls einen Iterationsschritt ausführen. Dabei entscheidet jede Blüte, ob es an der Zeit ist einen Duftstoff abzugeben. Wenn dies der Fall ist, erzeugt die Blüte einen Duftstoff und beauftragt ihre Pflanze über die Methode *propagateFragrance(Fragrance fragrance, int env)* diesen Duftstoff an alle umliegenden Felder zu propagieren. Dabei werden nur Felder berücksichtigt, die mit dem Feld der Pflanze direkt verbunden sind. Der Duftstoff wird bei einer Propagation bei den jeweiligen Feldern angemeldet, so dass die Datenhaltung der Duftstoffe in die Zuständigkeit der Felder fällt.

Feld

Ein Feld hat u.a. die Aufgabe, die auf ihm abgelegten Duftstoffe zu verwalten. Dazu zählt die Veränderung der Intensität wie auch die Verbreitung der Duftstoffe an Nachbarfelder. Damit ein Bestäuber einem, auf dem Feld geschnupperten, Duftstoff folgen kann, muss das Feld die Duftstoffe zusammen mit ihren Herkunftsfeldern abspeichern.

Die künstliche Blütenbestäubung betrachtet einen Duftstoff als Nachricht, deren Aktualität im Laufe der Zeit abnimmt. Die Aktualität einer Nachricht entspricht dabei der Intensität des Duftstoffes. Demnach verringert ein Feld in jedem Iterationsschritt die Intensität seiner Duftstoffe, da seit dem letzten Iterationsschritt Zeit vergangen ist. Des Weiteren muss ein Feld während jeder

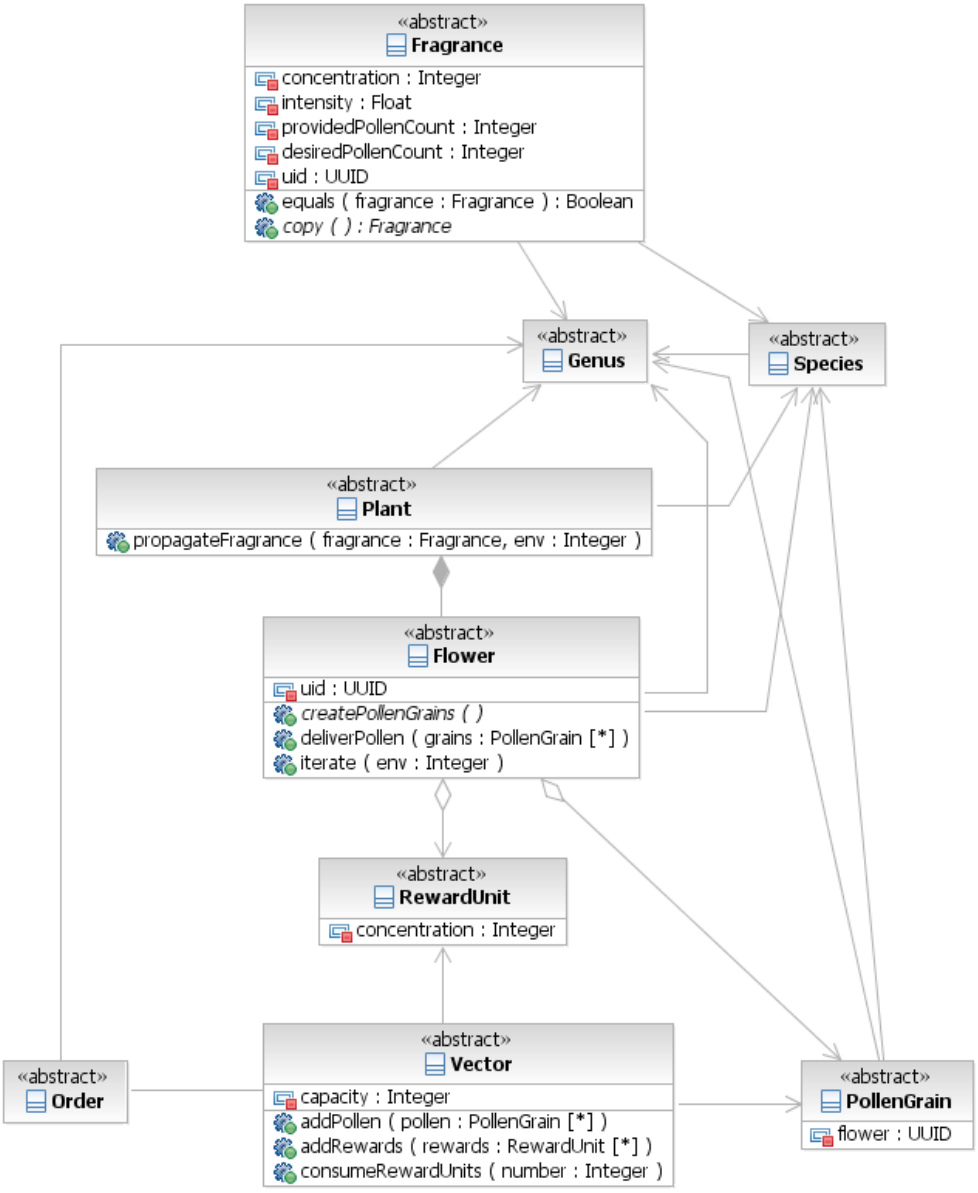

Abbildung 4.6: Designklassendiagramm des Bestäubungsmodells.

Iteration die auf ihm abgelegten Duftstoffe an seine Nachbarfelder propagieren. Dafür werden sie bei den jeweiligen Nachbarfeldern angemeldet. Bei der Anmeldung eines Duftstoffes auf einem Feld ist es wichtig, dass nur der jeweils aktuellste Duftstoff, der von einer Blüte abgegeben wurde, auf dem Feld bleibt. Aus diesem Grund überprüft jedes Feld bei einem Anmeldeversuch eines Duftstoffes, unabhängig, ob von einer Pflanze oder einem Nachbarfeld, ob ein Duftstoff von der gleichen Blüte bereits auf dem Feld liegt. Falls der anzumeldende Duftstoff aktueller ist, wird der auf dem Feld bereits liegende Duftstoff gelöscht und der aktuellere angemeldet. Bei der Verbreitung eines Duftstoffes muss davon ausgegangen werden, dass sich der Duftstoff immer weiter von seiner Blüte entfernt. Demnach verlieren seine Informationen durch den größeren Abstand zu der Blüte zusätzlich an Aktualität. Dieser Verlust wird ebenso von den Feldern der Umgebung durch die Verringerung der Intensität des Duftstoffes umgesetzt.

Bestäuber

Wie bereits in Abbildung 2.3 zu sehen war, ist das Verhalten eines Bestäubers während eines Iterationsschrittes sehr komplex. Ein Bestäuber befindet sich zu Beginn eines Iterationsschrittes auf einem Feld x und möchte auf ein für ihn möglichst gutes Feld gehen. Dafür fragt er bei Feld x alle für ihn relevanten Duftstoffe nach und evaluiert diese anhand einer Nutzenfunktion. Die Nutzenfunktion $N(f)$ ist folgendermaßen aufgebaut:

Falls der Bestäuber keine Pollen geladen hat: Ist ein Bestäuber nicht beladen, so spielen für die Nutzenfunktion die Intensität und die Konzentration des Duftstoffes wie auch die Anzahl an ladbaren Pollen eine Rolle. Sollte der Bestäuber eine geringere Kapazität haben, als die Blüte Pollen anbietet, so wird nur die Kapazität des Bestäubers in der Nutzenfunktion berücksichtigt. Sollte hingegen die Kapazität größer als die Anzahl angebotenen Pollen sein, so fließt die Anzahl in die Berechnung ein. Folgend ist die Nutzenfunktion $N(f)$ eines unbeladenen Bestäubers, auf Basis des in Abschnitt 2.3 definierten Bestäubungsmodells, dargestellt:

$$N(f) = x * f(i) * f(c)$$

$$x = if(cap < f(k))\ then\ cap\ else\ f(k)$$

Falls der Bestäuber Pollen geladen hat: Ist ein Bestäuber teilweise oder voll beladen, so sind für die Nutzenfunktion die Intensität und die Konzentration des Duftstoffes wie auch die ladbaren und ablegbaren Pollen relevant. Sollte ein Bestäuber mehr Pollen geladen haben, als die Blüte benötigt, so wird die Anzahl der benötigten Pollen, sonst die Anzahl der geladenen Pollen in der Berechnung verwendet. Sollte des Weiteren der Bestäuber zusätzlich mehr Pollen laden können als die Blüte anbietet, so fließt nur die Anzahl der angebotenen Pollen, sonst die Anzahl der noch aufnehmbaren Pollen in die Nutzenfunktion. Die Nutzenfunktion $N(f)$ eines teilweise bzw. vollständig beladenen Bestäubers ist wie folgt definiert:

$$N(f) = (y + z) * f(i) * f(c)$$

$$y = if((cap - |VP|) < f(k))\ then\ cap - |VP|\ else\ f(k)$$

$$z = if(|VP| < f(l))\ then\ |VP|\ else\ f(l)$$

Auf Basis der Ergebnisse der Nutzenfunktion entscheidet sich der Bestäuber für das nächste Feld. Auf dem neuen Feld überprüft er nun, ob sich genau die Blüte auf dem Feld befindet, deren Duftstoff er gefolgt ist. Sollte dies der Fall sein, werden Pollen ausgetauscht, d.h. er gibt Pollen ab bzw. nimmt sie auf, und erhält Belohnungseinheiten für den Anflug auf die Blüte. Sollte er Pollen abgeben, so werden die Belohnungen, die er bekommt, gleichzeitig mit den Belohnungen, die er für die Aufnahme der Pollen bekam, konsumiert. Sollte sich jedoch keine Blüte auf dem Feld befinden, so ist der Iterationsschritt für den Bestäuber beendet.

4.3.3 Graphische Benutzeroberfläche

Die graphische Benutzeroberfläche ermöglicht dem Benutzer eine einfachere Nutzung der Simulationsumgebung. Demnach bietet das ValuePane die Möglichkeit, alle relevanten Informationen, wie z.B. die Intensität eines Duftstoffes oder die Kapazität eines Bestäubers zu verändern. Hat sich der Benutzer für ein Modell, ein Szenario und eine Konfiguration entschieden, lädt er diese. Dieser mehrere Aufgaben umfassende Vorgang wurde bereits in Abschnitt 4.3.1 genauestens beschrieben. Nach einem erfolgreichen Ladevorgang wird der DrawingArea eine Referenz auf die SimulationsObjekte gewährt, die diese zur Darstellung der Zustände der Simulation verwendet. Für eine regelmäßige Aktualisierung des Simulationszustands wird die DrawingArea über das Observer-Pattern von dem Simulator informiert, d.h. nach jedem Iterationsschritt informiert der Simulator seine Überwacher, darunter auch die DrawingArea, die auf Basis der Benachrichtigung die Anzeige der SimulationsObjekte aktualisiert.

4.3.4 Datenbankanbindung

Die Klasse DBStore verhält sich ähnlich zu der Klasse DrawingArea. DBStore wird vor einem Start des Simulators von dem Controller angelegt. Dabei wird der Klasse eine Referenz auf alle SimulationsObjekte übergeben, deren Veränderungen in der Datenbank gespeichert werden sollen. Des Weiteren meldet der Controller den DBStore bei der Klasse Simulator an, damit dieser über das Ende jeder Iteration informiert wird. Nach jeder Benachrichtigung speichert die Klasse DBStore für die Evaluation relevante Informationen in eine Datenbank.

4.4 Realisierung

Die Realisierung eines Systems entspricht der konkreten Implementierung. In diesem Abschnitt sollen jedoch keine Codefragmente beschrieben, sondern eine Übersicht über implementierungsspezifische Details gegeben werden. So wurde im Laufe der Umsetzung klar, dass einige Teile des Systems nur durch eine komplizierte Realisierung effizienter gemacht werden können. Neben einer Beschreibung der Realisierung stellt dieser Abschnitt zusätzlich die Umsetzung der graphischen Benutzeroberfläche vor.

4.4.1 Verbreitung von Duftstoffen

Ein Duftstoff beinhaltet Informationen über eine Blüte, die Pollen anbietet oder nachfrägt, in Zusammenhang mit einer Intensität, die die Aktualität der Informationen bewertet. Der Duft-

stoff hält jedoch keine Informationen, von welcher Blüte er stammt bzw. aus welcher Richtung er kommt. Aus diesem Grund müssen sich die Felder der Umgebung das Herkunftsfeld eines neuen Duftstoffes merken. Da jedes Feld maximal 8 Nachbarn besitzt, konnte für jeden Nachbarn ein "Sammelbehälter"(bucket) erstellt werden, in dem die Duftstoffe gehalten werden. Sollte ein Duftstoff beispielsweise von Norden auf ein neues Feld kommen, so wird seine Referenz in dem ersten Behälter gespeichert. Dadurch kann nachvollzogen werden, von welchem Nachbarfeld ein Duftstoff kommt.

Bei der Verbreitung eines Duftstoffes wird bei allen Nachbarfeldern eine Kopie des zu verbreitenden Duftstoffes angemeldet. Dies hat zur Folge, dass der ursprüngliche Duftstoff auf seinem Feld bleibt und bei der nächsten Iteration eventuell noch einmal propagiert wird. Diese Verbreitung ist jedoch nicht notwendig, weil alle Nachbarfelder den Duftstoff bereits besitzen. Um diese überflüssigen Berechnungen zu vermeiden, speichert jedes Feld zu jedem Duftstoff ein flag, ob dieser bereits propagiert wurde oder nicht. Dadurch kann vor einer Propagation eines Duftstoffes überprüft werden, ob die Verbreitung noch nötig ist.

Im Zusammenhang mit der Verwaltung der Duftstoffe, muss ein Feld immer gewährleisten, dass es nie zwei Duftstoffe der gleichen Blüte gespeichert hat. Diese Invariante wird dadurch erfüllt, dass sobald sich ein Duftstoff bei einem Feld anmeldet, dieses überprüft, ob ein Duftstoff der gleichen Blüte bereits gespeichert ist. Ist dies der Fall wird die Intensität beider Duftstoffe verglichen und der aktuellere Duftstoff bevorzugt gespeichert. Falls kein anderer Duftstoff der gleichen Blüte gespeichert ist, kann der neu angemeldete Duftstoff gespeichert werden. Um Duftstoffe auf "Gleichheit" in Bezug auf ihre Herkunftsblüte zu vergleichen, wird jedem Duftstoff bei dessen Erzeugung die eindeutige Bezeichnung der Blüte als Attribut mitgegeben. Dadurch ist ein Duftstoff einer Blüte eindeutig zugeordnet.

Der wohl wichtigste Punkt bei der Verbreitung der Duftstoffe hängt jedoch mit dem sequentiellen Ablauf der Simulation zusammen. Der Simulator gibt in einer Iteration jedem SimulationsObjekt die Möglichkeit einmal zu iterieren. Dadurch entsteht eine Reihenfolge, nach der die einzelnen SimulationsObjekte abgearbeitet werden. Beispielsweise kann Feld x seine Duftstoffe unter anderem an Feld y propagieren, welches noch keinen Iterationsschritt durchgeführt hat. Bekommt Feld y eine Zeiteinheit um zu iterieren, werden die neu hinzugekommenen Duftstoffe innerhalb der gleichen Iteration erneut propagiert und damit ihre Intensität reduziert. Um dieses Verhalten zu vermeiden, ist es notwendig, dass ein Feld zwei "Zustände" der gespeicherten Duftstoffe verwaltet. Dadurch kann zwischen dem neuen und dem alten, noch zu propagierenden Zustand unterschieden werden.

4.4.2 Aufbau einer *.env-Datei

Wie in Abschnitt 4.3.1 beschrieben wurde, kann die Klasse EnvironmentLoader eine Gridfeld-Umgebung aus einer Datei laden. Da es sich um eine Umgebungsdatei handelt, muss die Datei die Endung '.env' besitzen.
Eine *.env-Datei, die eine 5x5 Umgebung repräsentiert, kann wie links in Tabelle 4.1 dargestellt, aufgebaut sein. Die '1' entspricht einem Feld, die Zeichen '-', 'x', '/', '\'und '|' entsprechen den Verbindungen zwischen den Feldern. Die rechte Seite der Tabelle 4.1 stellt die aus der Datei resultierende Umgebung dar. Durch die vorhandenen bzw. nicht vorhandenen Verbindungen wird festgelegt, wie sich ein Duftstoff verbreitet und wie sich ein Bestäuber in der Umwelt bewegen kann.

```
1 - 1 - 1 - 1 - 1
| x | x | x | x |
1 - 1 - 1 - 1 - 1
        |
1 - 1     1 - 1 - 1
| x |     | x | x |
1 - 1 - 1 - 1 - 1
| x |     | x | x |
1 - 1     1 - 1 - 1
```

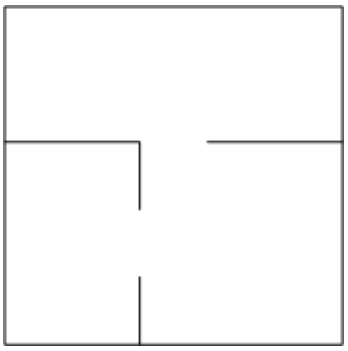

Tabelle 4.1: Darstellung einer *.env-Datei und der daraus resultierenden Umgebung

4.4.3 Graphische Benutzeroberfläche

Eine graphische Benutzeroberfläche (GUI) bietet dem Anwender der Simulationsumgebung eine einfache Benutzung und Handhabung des Systems. Dies gilt jedoch nicht generell für jede Oberfläche, da diese nicht immer von jedem Anwender als einfach empfunden wird. Wie gut ein Benutzer eine Oberfläche bedienen kann, hängt von seiner Übung und seinem Verständnis für das Problem ab. Bei dem Design einer Oberfläche muss man sich immer darüber im Klaren sein, welcher Benutzergruppe die GUI zur Verfügung gestellt werden soll. Die Oberfläche der Simulationsumgebung für selbst-organisierende Systeme orientiert sich in erster Linie an ungeübten Benutzern. So führt sie den Anwender durch eine Reihe an auszuwählenden Informationen, die für die Simulation relevant sind, ohne ihn durch eine Fülle an Möglichkeiten zu “erschlagen”.
Startet der Benutzer die Simulationsumgebung, erscheint eine graphische Benutzeroberfläche ähnlich der in Abbildung 4.7, jedoch ohne die Darstellung der Zeichenebene. Zu Beginn muss der Benutzer eine Konfiguration wählen, die durch ein Modell, ein Szenario und eine Konfigurationsart spezifiziert ist. Dafür bietet die Oberfläche anfangs nur die Möglichkeit ein passendes Modell zu wählen, so dass der Benutzer durch keine andere Eingabemöglichkeit abgelenkt ist. So kann er seine ganze Konzentration auf das momentane Problem lenken und ist nicht durch eine Fülle an auszufüllenden Feldern bzw. auszuwählenden Optionen überfordert. Nachdem der Anwender nun ein Modell gewählt hat, wird er weiter durch eine Auswahl an wichtigen Informationen der Simulation geführt. Für den genauen Ablauf der Führung durch die Simulationsumgebung sei der Leser hiermit an das Programm selbst bzw. an Anhang A verwiesen.

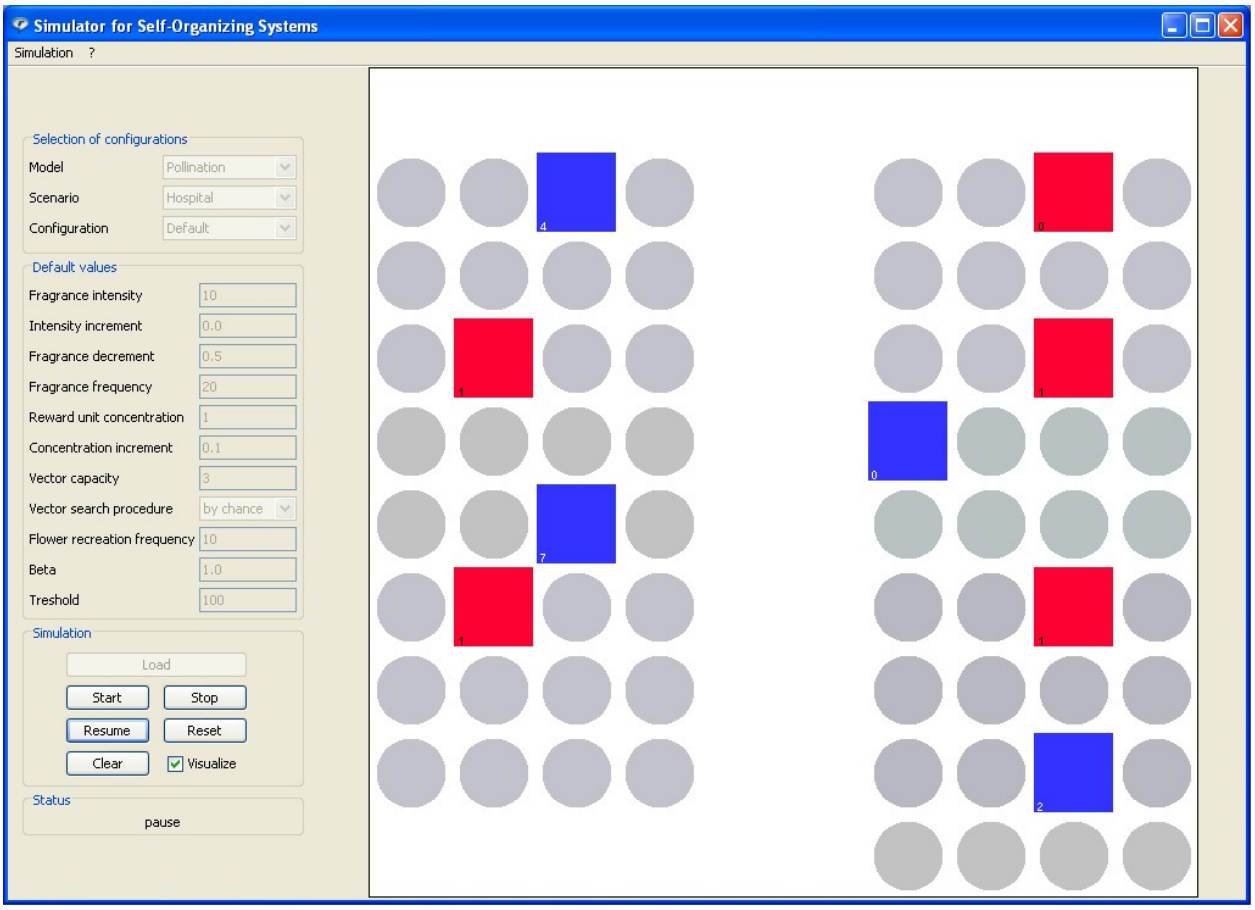

Abbildung 4.7: Graphische Oberfläche des Simulator für selbst-organisierende Systeme.

Kapitel 5

Evaluation

Für die Evaluation des Modells der künstlichen Blütenbestäubung werden verschiedene Szenarien unter Berücksichtigung der in Abschnitt 3.1 festgelegten Evaluationskriterien bewertet. Das dafür benötigte Bewertungsschema wird in 5.1 aufgestellt. Basis jedes Simulationsexperiments ist ein Gridfeld bestehend aus 10x10 Feldern, welches an den Rändern begrenzt, aber intern voll verbunden ist. D.h. jedes Feld hat genau 8 Nachbarn, außer es handelt sich um ein Randfeld. Nachdem der Ablauf eines Simulationsexperiments nicht deterministisch ist, werden für die Bewertung einer Konfiguration mindestens 100 Simulationsexperimente mit den gleichen Startvorraussetzungen und Parametereinstellungen durchgeführt.

Da das Modell bisher nur formal vorlag, wird in Abschnitt 5.2 an einem sehr einfachen Beispiel ein erster funktionaler Test durchgeführt. Dieser wird Aufschluss darüber geben, ob das reale Verhalten des formalen Modells dem gewünschten Verhalten entspricht. In den darauf folgenden Abschnitten werden u.a. die Skalierbarkeit, Robustheit und Flexibilität des Modells betrachtet, bevor in Abschnitt 5.6 der simulierte Einsatz der künstlichen Blütenbestäubung in einem Krankenhaus bewertet wird.

5.1 Bewertungsschema

Um die in Abschnitt 3.1 vorgestellten Evaluationskriterien zu messen, muss ein Vergleich verschiedener APS möglich sein. Im Folgenden wird ein Bewertungsschema für die verschiedenen Bestäubungssysteme benötigt.

Das Bewertungsschema für die Evaluation innerhalb dieser Arbeit verwendet für einen Vergleich zweier APS deren Effektivität. Die Effektivität eines APS entspricht dem Nutzen-Kosten-Verhältnis der Simulation. Die Effektivität wird durch folgende Werte ermittelt:

1. Die Dauer (n) einer Simulation. Je länger eine Simulation benötigt, desto uneffektiver ist das System.

2. Die Auslastung des Systems bzw. die Auslastung der Bestäuber des Systems. Je besser die Auslastung ist, desto effektiver ist das System. Die durchschnittliche Auslastung (av_util_{APS})

wird wie folgt berechnet:

$$av\_util_{APS} = \frac{1}{|V_{APS}| * n} * \sum_{t=1}^{n} \sum_{i=1}^{|V_{APS}|} util_{APS}(v_i, t)$$

$$util_{APS}(v, t) = \frac{|VP_{APS}(v, t)|}{cap_{APS}(v)}$$

wobei

$\|VP_{APS}(v,t)\|$	=	*Anzahl der geladenen Pollen des Bestäubers v in Iteration t*
$\|V_{APS}\|$	=	*Anzahl der Bestäuber des APS*
n	=	*Dauer der Simulation des APS*

3. Die Bestäubungsfrequenz einer Blüte, d.h. wie oft wird eine Blüte durchschnittlich in einer Iteration bestäubt. Je öfter eine Blüte in einer Iteration bestäubt wird, desto besser ist die Effektivität des Systems. Wird der Kehrwert gebildet, so gibt dieser die Bestäubungsperiode an, d.h. wie viele Iterationen eine Blüte durchschnittlich warten muss, bis sie bestäubt wird. Je länger eine Blüte warten muss, desto uneffektiver ist das System. Die Bestäubungsfrequenz (av_pol_{APS}) wird folgendermaßen berechnet:

$$av\_pol_{APS} = \frac{1}{|P_{APS}|} * \sum_{k=1}^{|P_{APS}|} (\frac{1}{|F_{APS}|_{p_k}} * \sum_{j=1}^{|F_{APS}|_{p_k}} pol_{APS}(f_{p_k j}))$$

$$pol_{APS}(f) = \frac{\sum_{t=1}^{n} isMet_{APS}(f, t)}{lastIteration_{APS}(f)}$$

wobei

$\|P_{APS}\|$	=	*Anzahl der Pflanzen des APS*
$\|F_{APS}\|_{p_k}$	=	*Anzahl der Blüten des APS unter der Bedingung, dass sie von Pflanze p_k stammen*
$f_{p_k j}$	=	*die j-te Blüte von Pflanze p_k*
$isMet_{APS}(f,t)$	=	*1, falls Blüte f in Iteration t bestäubt wurde* *0, andernfalls*
$lastIteration_{APS}(f)$	=	*die letzte Iteration, in der die Blüte f Pollen nachfrägt bzw. anbietet*
n	=	*Dauer der Simulation des APS*

4. Die Anzahl an propagierten Duftstoffen in der Umwelt. Je länger eine Blüte warten muss, desto mehr Duftstoffe sendet sie aus. Je mehr Duftstoffe pro Iteration in der Umwelt liegen, desto stärker ist der Verbreitungsaufwand und desto uneffektiver ist das System. Die Anzahl der Duftstoffe pro Iteration wird über die Felder der Umgebung wie folgt berechnet:

$$av\_frag_{APS} = \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{\#rows} \sum_{j=1}^{\#columns} pollenCount(field_{ij}, t)$$

wobei

$pollenCount(field_{ij}, t)$ = *Anzahl der Pollen auf* $field_{ij}$ *in Iteration t*
$field_{ij}$ = *Feld der Stelle [i,j] im Grid*
n = *Dauer der Simulation des APS*

Der Nutzen eines System liegt einerseits in dessen Auslastung wie auch in dessen Bestäubungsfrequenz. Bei einem Vergleich zweier APS sollte das APS mit der besseren Auslastung der Bestäuber wie auch der höheren Bestäubungsfrequenz als das Bessere der beiden Systeme identifiziert werden. Demnach ist die Effektivität im Zähler durch Auslastung * Bestäubungsfrequenz definiert. Die Kosten eines Systems bestehen aus den pro Iteration in der Umwelt liegenden Duftstoffen wie auch aus der Dauer des Systems bis zu dessen Terminierung. Bei einem Vergleich zweier APS sollte das APS mit der geringeren Anzahl an Duftstoffen wie auch der geringeren Dauer als besseres System bewertet werden. Die Effektivität(*Effect*) eines Systems wird demnach wie folgt berechnet:

$$Effect_{APS} = \frac{av\_util_{APS} * av\_pol_{APS}}{n * av\_frag_{APS}}$$

Basierend auf der hiermit definierten Effektivität eines APS werden in den folgenden Abschnitten Vergleiche zwischen verschiedenen Bestäubungssystemen angestellt. Dabei ist darauf zu achten, dass nicht jedes beliebige System mit einem anderen vergleichbar ist. Nur unter der Voraussetzung, dass alle Parameter wie auch die Ausgangssituation der Blüten und Bestäuber identisch ist, kann in Bezug auf einen zu betrachtenden Wert auf Basis der Effektivität evaluiert werden. Demnach bei zwei zu vergleichenden Systemen alle Werte bis auf den Vergleichswert identisch bleiben.

5.2 Funktionalität der Simulation

Bevor das formale Modell der Blütenbestäubung an einem bzw. mehreren Szenarien evaluiert werden kann, muss dessen Verhalten auf seine Richtigkeit geprüft werden. Zu diesem Zweck wird in Abschnitt 5.2.1 ein einfaches Simulationsmodell erstellt. Auf Basis der Simulationsergebnisse dieses Modells (siehe Abschnitt 5.2.2) werden in Abschnitt 5.2.3 Verbesserungen für das Verhalten des Systems vorgeschlagen.

5.2.1 Einfaches Simulationsmodell

Für einen ersten Funktionstest wird ein einfaches Simulationsmodell auf Basis des in Abschnitt 2.3 beschriebenen formalen Modells erstellt.

Gattungen:	g_1	**Ordnungen:**	o_1
Arten:	s_1	**Regeln:**	$s_1 \rightarrow g_1$
Pflanzen:	p_1, p_2		$o_1 \rightarrow g_1$
Blüten:	f = *(g, s, p,* $\#Pol^{prov}$*,* $\#Pol^{des}$*,c, i)*	**Bestäuber:**	v = *(o, cap, VR, VP)*
	$f_1 = (g_1, s_1, p_1, 20, 20, 3, 10)$		$v_1 = (o_1, 6, \emptyset, \emptyset)$
	$f_2 = (g_1, s_1, p_2, 20, 20, 3, 10)$		

Tabelle 5.1: Einfaches Simulationsmodell.

Das in Tabelle 5.1 dargestellte Modell besteht aus einer Gattung g_1, einer Art s_1 und einer Ordnung o_1. Die Regeln besagen, dass s_1 der Gattung g_1 angehört und Bestäuber der Ordnung o_1 Blüten der Gattung g_1 anfliegen. Des Weiteren beinhaltet das Modell zwei verschiedene Blüten, die jeweils 20 Pollen anbieten wie auch nachfragen. Dabei sollten auf Grund der gleichen Art die von Blüte f_1 angebotenen Pollen zu Blüte f_2 transportiert werden. Die von Blüte f_1 nachgefragten Pollen müssen von Blüte f_2 stammen. Des Weiteren besitzt das Modell einen Bestäuber, der in der Lage ist, maximal sechs Pollen von jeder Blüte zu transportieren.

Um dieses formale Modell simulieren zu können, wird noch eine Umgebung (nicht zu verwechseln mit der Simulationsumgebung) benötigt, in der sich die Bestäuber frei bewegen können. Wie zu Beginn dieses Kapitels erwähnt, wird das Modell in jedem Simulationsexperiment auf einem Gridfeld der Größe 10x10 ohne Barrieren, aber mit Rändern evaluiert. Der Aufbau eines einfachen Simulationsexperiments mit den Komponenten des einfachen Simulationsmodells ist in Abbildung 5.1 dargestellt.

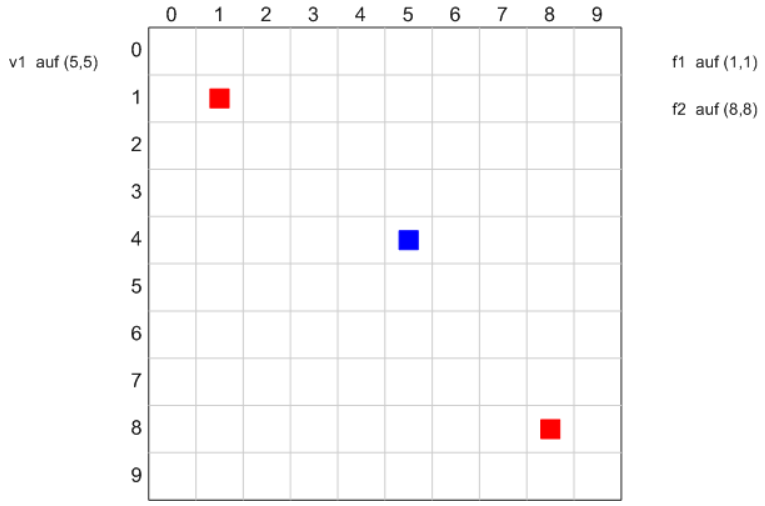

Abbildung 5.1: Ein einfaches Simulationsexperiment.

Basierend auf diesen Vorgaben wird das formale Modell nun simuliert.

5.2.2 Simulationsergebnisse

Nach einer ersten Simulation fällt auf, dass jede Blüte bestäubt wurde, jedoch u.a. durch Pollen, die sie selbst zur Verfügung gestellt hat. Dies hängt mit dem in Abschnitt 2.3 beschriebenen Verhalten eines Bestäubers zusammen. Demnach sucht der Bestäuber einen Duftstoff, dem er folgen kann. Wurde ein passender Duftstoff gefunden, versucht der Bestäuber diesem bis zu seiner aussendenden Blüte zu folgen. Ist er bei der Blüte angekommen, gibt er passende Pollen ab und nimmt passende Pollen auf. In diesem Simulationsmodell bietet z.B. die Blüte f_1 Pollen an und fragt gleichzeitig Pollen der gleichen Art und Gattung nach. Ein Bestäuber folgt einem Duftstoff bis zu Blüte f_1, nimmt Pollen auf und versucht im nächsten Schritt erneut einen geeigneten Duftstoff zu finden. Dabei trifft er auf den noch nicht vollständig evaporierten Duft vom f_1, folgt diesem zur Blüte und gibt dort seine geladenen Pollen ab. Dadurch ist eine Fremdbestäubung nur dann möglich, wenn

sich der Bestäuber in dem Augenblick von der Blüte löst, in dem alle alten Duftstoffe verschwunden sind und noch keine neuen Duftstoffe ausgesendet wurden. Da Selbst-Bestäubung in dem Modell der künstlichen Blütenbestäubung unerwünscht ist, werden folgend verschiedene Alternativen zur Vermeidung dieses Problems vorgestellt und auf ihre Einsatzfähigkeit geprüft.

5.2.3 Alternatives Verhalten

Eine Vermeidung der Selbst-Bestäubung kann einerseits durch Änderung des statischen Modells wie auch des dynamischen Verhaltens der Bestäuber bzw. Blüten erreicht werden.

1. **Veränderung des statischen Modells:** Eine mögliche Alternative um Selbst-Bestäubung zu vermeiden, wäre eine Änderung des Simulationsmodells. Demnach hätte jede Pflanze anstatt einer Blüte, die gleichzeitig Pollen anbietet und nachfrägt, jeweils zwei Blüten, eine zum Anbieten und eine zum Nachfragen von Pollen. Die beiden Blüten einer Pflanze müssen des Weiteren von verschiedenen Arten sein, da sonst eine ungewollte Bestäubung zwischen ihnen stattfindet. Das neue Simulationsmodell ist in Tabelle 5.2 dargestellt.

Gattungen:	g_1	**Ordnungen:**	o_1
Arten:	s_1, s_2	**Regeln:**	$s_1 \rightarrow g_1$
Pflanzen:	p_1, p_2		$s_2 \rightarrow g_1$
Blüten:	$f_{1a} = (g_1, s_1, p_1, 20,\ 0, 3, 10)$		$o_1 \rightarrow g_1$
	$f_{1b} = (g_1, s_2, p_1,\ 0, 20, 3, 10)$	**Bestäuber:**	$v_1 = (o_1, 6, \emptyset, \emptyset)$
	$f_{2a} = (g_1, s_2, p_2, 20,\ 0, 3, 10)$		
	$f_{2b} = (g_1, s_1, p_2,\ 0, 20, 3, 10)$		

Tabelle 5.2: Einfaches erweitertes Simulationsmodell.

   Während der Simulation des neuen Simulationsmodells wurde keine Blüte durch Pollen von sich selbst bestäubt. Die durchgeführten Veränderungen bzw. die Erweiterung des Modells um zusätzliche Blüten führt zu einem Ansteigen des benötigten Speicherplatzes, da für (fast) jede bisherige Blüte eine zweite Blüte hinzu kommt, die ebenfalls in regelmäßigen Abständen Duftstoffe abgibt. Eine Veränderung des Verhaltens des dynamischen Modells würde demnach weniger zusätzliche Ressourcen benötigen.

2. **Veränderung des dynamischen Modells:** Neben einer statischen Modelländerung kann auch der dynamische Teil, d.h. das Verhalten der Bestäuber bzw. der Blüten, für die Vermeidung von Selbst-Bestäubung in Betracht gezogen werden. Basierend auf dem Verhalten eines Bestäubers (siehe Abbildung 2.3) kann eine Veränderung des dynamischen Modells einerseits die Auswertung der "geschnupperten" Duftstoffe auf einem Feld und andererseits die Abgabe der Pollen an eine Blüte betreffen. Beide Aktionen werden in den folgenden Abschnitten verändert.

   1. **Merken der besuchten Blüten:** Da der Bestäuber die Pollen von einer Blüte zu einer anderen transportiert, kann durch Erweiterung seines Wissens eventuell Selbst-Bestäubung vermieden werden. Dieses zusätzliche Wissen besteht aus den angeflogenen Blüten, d.h. nach jeder Aufnahme von Pollen merkt sich der Bestäuber, welche Blüte er

gerade angeflogen hat. Diese Blüten werden in dem aktuellen Sammelflug des Bestäubers nicht mehr angeflogen. Sobald ein Bestäuber keine Pollen mehr geladen hat, kann er seine gespeicherten Informationen verwerfen und es beginnt ein neuer Sammelflug.

Bei einer Veränderung des Verhaltens des Bestäubers muss nicht nur auf die Aufnahme und Abgabe der Pollen an geeignete Blüten geachtet werden. Auch das Verhalten, welchen Duftstoffen ein Bestäuber folgt, muss angepasst sein. So sollte keinesfalls einem Duftstoff gefolgt werden, dessen Blüte bereits von dem Bestäuber gemerkt wurde.

Eine Simulation des Modells zeigt, dass durch die Veränderung des Verhaltens der Bestäuber eine Selbst-Bestäubung der Blüten ausgeschlossen wurde.

2. **Intelligente Abgabe von Pollen:** Die Pollenabgabe wird neben dem Bestäuber auch entscheidend von der Blüte durchgeführt. So hat eine Blüte die Eigenschaft, sich auf den Bestäuber zu verlassen und bis zu einer gesättigten Menge jedes Pollenkorn anzunehmen, dass dieser anbietet. Dieses Verhalten kann jedoch dahingehend abgeändert werden, dass eine Blüte selbst darauf achtet, bei der Annahme keine eigenen Pollen entgegen zu nehmen.

   Demnach kann ein Bestäuber nun jede beliebige Blüte anfliegen, da die Blüten selbst auf eine Vermeidung der Selbst-Bestäubung achten. Jedoch muss ein Bestäuber sein Verhalten in Bezug auf "geschnupperte" Duftstoffe ändern, d.h. er darf nicht durchgehend von einer Blüte angezogen werden, die keine Pollen mehr von ihm annimmt, da alle zur Verfügung stehenden Pollenkörnern von ihr stammen. Dieses Verhalten kann dadurch umgesetzt werden, dass ein Bestäuber regelmäßig überprüft, ob er Pollenkörner von nur einer Blüte transportiert. Ist dies der Fall, wird dem Duftstoff dieser Blüte nicht gefolgt. Transportiert der Bestäuber hingegen mehrere Pollenkörner verschiedener Blüten, so kann er jedem passenden Duftstoff folgen, da er mindestens ein fremdes Pollenkorn abgeben kann.

   Eine Simulation des eben beschriebenen Modells zeigt, dass auch diese Veränderung mögliche Selbst-Bestäubungen einzelner Blüten ausschließt.

3. **Erweiterung der intelligenten Abgabe von Pollen:** Durch die Simulation des Modells der intelligenten Abgabe von Pollen ist aufgefallen, dass ein Bestäuber nach Aufnahme der Pollen einer bestimmten Blüte zu Beginn immer wieder zu der besuchten Blüte zurückkehrt. Dieses Verhalten kommt auf Grund veralteter Duftstoffe auf den umliegenden Feldern zustande. Um das Modell effizienter zu gestalten, muss der Bestäuber Wissen über die zuletzt besuchte Blüte speichern. Dadurch werden veraltete Duftstoffe einer Blüte nach Aufnahme deren Pollen ignoriert und der Bestäuber kann sich sofort wieder auf die Suche nach neuen Blüten und deren Duftstoffen machen.

   Eine Simulation des Modells zeigt, dass die Erweiterung des Wissens eines Bestäubers keine Veränderung im Bezug auf die Selbst-Bestäubung zur Folge hat.

3. **Gegenüberstellung:** Um festzustellen, welche Alternative für das Modell der künstlichen Blütenbestäubung am geeignetsten ist, wird eine Gegenüberstellung der Ansätze durchgeführt. Dafür wird jede Alternative auf Basis des einfachen Simulationsmodell 100 mal simuliert, um anhand der durchschnittlichen Dauer eines Simulationsexperiments Schlüsse über sie ziehen zu können.

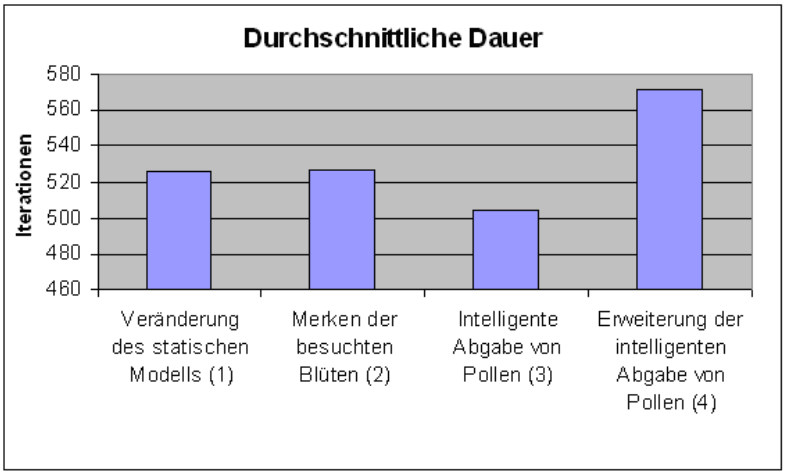

Abbildung 5.2: Durchschnittliche Simulationsdauer

Wie in Abbildung 5.2 zu erkennen ist, benötigt Alternative 3 durchschnittlich am wenigsten lang, Alternative 4 hingegen durchschnittlich am längsten. Dieses Ergebnis ist überraschend, da Alternative 4 nur eine Erweiterung von Alternative 3 darstellt und diese in keiner Weise einschränkt. Der Grund dieses Ergebnisses ist die fest eingestellte Intensität eines Duftstoffes während der Simulationsexperimente. Der Wert der Intensität ist so gewählt, dass die ausgesandten Duftstoffe einer Blüte nie die ganze Umgebung ausfüllen. Unter ungünstigen Voraussetzungen kann es dadurch dazu kommen, dass sich ein Bestäuber durch die zufällige Wahl des nächsten Feldes immer außerhalb der Reichweite der relevanten Blüte befindet. Sollte dies in einem Simulationsexperiment der Fall sein, so werden mehr Iterationen für eine Terminierung benötigt.

Da das einfache Simulationsmodell keine geeignete Grundlage für eine Unterscheidung der Alternativen bietet, wird nun versucht, ein passendes Modell zu erstellen, welches eine Bewertung der verschiedenen Möglichkeiten zur Vermeidung von Selbst-Bestäubung erlaubt. Für eine Bewertung der einzelnen Alternativen, muss das Modell mehrere Blüten (>2) besitzen, die Pollen der gleichen Art und Gattung abgeben und gleichzeitig nachfragen. Bei einem Modell mit nur zwei Blüten, ist eine Bewertung der Alternativen nicht aussagekräftig, da deren Vorteile durch die eingeschränkte Auswahl an Blüten nicht zur Geltung kommen. Besitzt ein Modell jedoch mehr als zwei Blüten, die gleichzeitig Pollen der gleichen Art und Gattung abgeben und annehmen, so ist immer die Wahrscheinlichkeit eines Deadlocks gegeben. Für eine Verdeutlichung des Problems soll Abbildung 5.3 betrachtet werden. Eine terminierende Flugroute eines Bestäubers wäre in diesem Modell $f_1 \rightarrow f_2 \rightarrow f_3 \rightarrow f_1$, da dadurch jede Blüte befruchtet wird und demnach die Simulation terminieren kann. Es gibt jedoch eine Möglichkeit wie auch dieses Modell in einen Deadlock geraten kann. Die Flugroute $f_3 \rightarrow f_2 \rightarrow f_1 \rightarrow f_2$ bewirkt, dass der Bestäuber mit 10 Pollen der Blüte f_1 beladen ist, diese aber nirgendwo abgegeben werden können. Blüte f_1 hingegen fragt genau 10 Pollen nach, kann jedoch auf Grund einer Selbst-Bestäubung die 10 Pollen des Bestäubers nicht entgegennehmen.

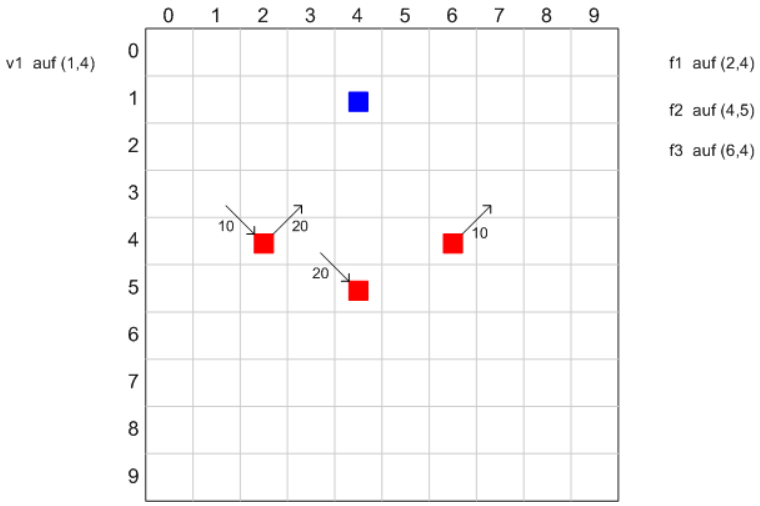

Abbildung 5.3: Problematisches Simulationsmodell.

Das Problem eines Deadlocks ist immer gegeben, sobald mehr als zwei Blüten Pollen der gleichen Art und Gattung gleichzeitig abgeben und annehmen. Die Vorteile der vier Alternativen gegen Selbst-Bestäubung kommen jedoch erst bei mehr als zwei Blüten der gleichen Art und Gattung zum tragen. Resultierend daraus, komme ich zu der Erkenntnis, dass die Wahl einer 'günstigsten' Alternative hinfällig ist, so lang das Problem des Deadlocks nicht gelöst wurde.

Trotz dieser Ergebnisse muss für die Simulationen im Rahmen dieser Arbeit ein einheitliches Verhalten gewählt werden. Demnach wird die intelligente Pollenabgabe (Alternative 3) für die folgenden Simulationen festgelegt.

5.3 Skalierbarkeit

Für die Skalierbarkeit eines Systems muss der zusätzliche Aufwand ermittelt werden, der bei Hinzufügen einer weiteren Modellkomponente entsteht. Relevante Modellkomponenten der künstlichen Blütenbestäubung sind die Bestäuber wie auch die Blüten. Basierend auf dem einfachen Simulationsmodell aus Abschnitt 5.2 können für die Skalierbarkeit die Kosten für einen zusätzlichen Bestäuber wie auch für eine zusätzliche Blüte ermittelt werden.

5.3.1 Veränderung der Menge an Bestäubern

Für die Bewertung des einfachen Simulationsmodells werden Simulationsexperimente mit einem bis zu zehn Bestäubern durchgeführt. Anhand des Bewertungsschemas aus 5.1 ergeben sich für die Effektivität die in Abbildung 5.4 dargestellte Kurve. Anhand dieser Kurve ist erkennbar, dass für das einfache Simulationsmodell eine Menge von vier Bestäubern am effektivsten ist.

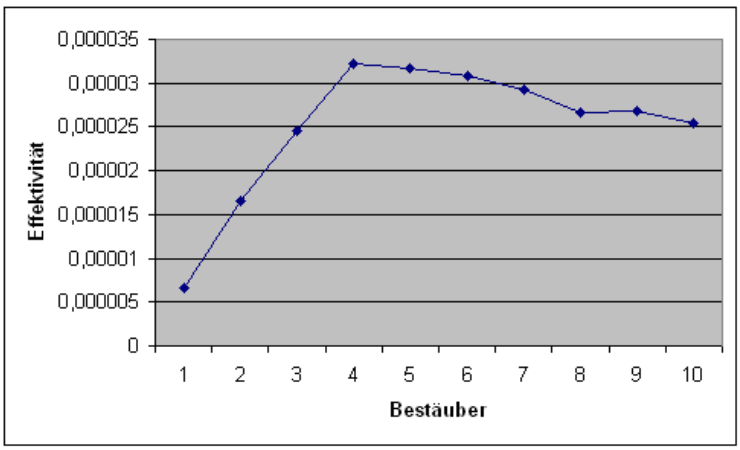

Abbildung 5.4: Die Effektivität des einfachen Simulationsmodells in Abhängigkeit von der Anzahl an Bestäubern.

Für eine Aussage, ob das System skalierbar ist, werden dessen Kosten benötigt. Die Kosten eines Systems sind laut Abschnitt 5.1 die in der Umwelt liegenden Duftstoffe wie auch die Terminierungsdauer des Systems. Bei einem Vergleich zweier APS ist das APS mit der größeren Anzahl an propagierten Duftstoffen wie auch der längeren Laufzeit bis zu dessen Terminierung als kostenintensiver einzustufen. Neben diesen beiden Werten muss jedoch berücksichtigt werden, dass auch die nicht genutzten Kapazitäten eines Bestäubers Kosten darstellen. Demnach fließt der Anteil der überschüssigen Kapazitäten in die Berechnung der Kosten ein. Bei einem Vergleich zweier Systeme, deren Auslastung identisch ist, muss jedoch auch die Anzahl an Bestäubern berücksichtigt werden. Bei einer Auslastung von 40% ist ein System mit 4 Bestäubern weniger kostenintensiv als ein System mit 6 Bestäubern. Aus diesem Grund muss die Anzahl der Bestäuber bei der Berechnung der Kosten ebenfalls berücksichtigt werden. Die entstandenen Kosten ($cost_{APS}$) eines Systems lassen sich wie folgt berechnen:

$$cost_{APS} = av\_frag_{APS} * n * (1 - av\_util_{APS}) * |V|$$

Analog zu der Effizienz eines Systems gilt auch hier, dass zwei Systeme in Bezug auf deren Kosten nur dann vergleichbar sind, wenn deren Einstellungen und Parameter bis auf einen veränderten Wert identisch sind.

Durch eine Veränderung der Menge an Bestäubern innerhalb des einfachen Simulationsmodells entsteht auf Basis der Kostendefinition die in Abbildung 5.5 dargestellte Kurve. Anhand dieser Kurve ist erkennbar, dass für jeden Bestäuber zusätzliche Kosten in Höhe von ca. 25000 hinzukommen (für eine Tabelle inklusive genauer Werte siehe Abbildung B.1). Das System ist demnach in Bezug auf seine Bestäuber gut skaliert, da die Kosten für jeden weiteren Bestäuber linear und nicht proportional ansteigen.

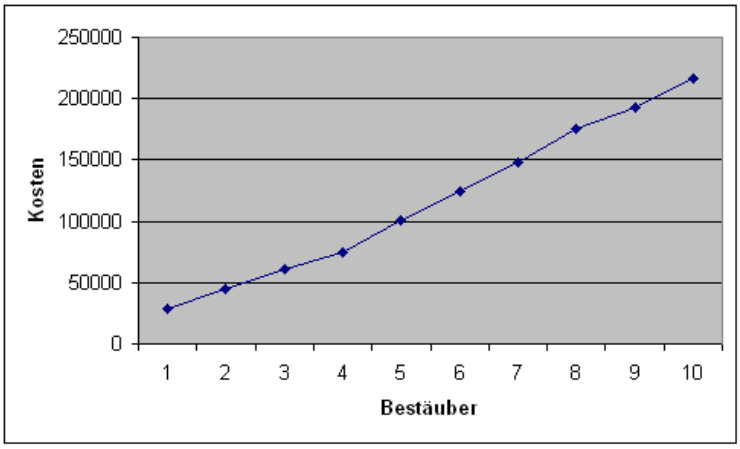

Abbildung 5.5: Kosten des einfachen Simulationsmodells bei steigender Anzahl an Bestäubern.

5.3.2 Veränderung der Menge an Blüten

Die Veränderung der Menge an Blüten innerhalb eines Modells ist um einiges problematischer als die Veränderung der Bestäuberanzahl. Das Hinzufügen einer neuen Blüte kann trivial dadurch erreicht werden, dass eine weitere Blüte, die weder Pollen anbietet noch nachfragt, zu Beginn der Simulation existiert. Diese Blüte wird jedoch bereits nach der ersten Iteration zerstört, so dass sie keinen relevanten Einfluss auf die Simulation hat. Um eine Blüte sinnvoll hinzuzufügen, muss diese in das Modell "eingebunden" werden, d.h. sie muss ebenfalls Pollen anbieten oder nachfragen. Die angebotene wie auch nachgefragte Menge muss jedoch mit dem bisherigen Modell in Einklang stehen, d.h. die gesamte Menge an angebotenen Pollen muss der gesamten Menge an nachgefragten Pollen entsprechen. Ist dies nicht der Fall, terminiert das System nicht.

Basierend auf dem einfachen Simulationsmodell kann eine dritte Blüte f_3 hinzugefügt werden, die ebenfalls 20 Pollen anbietet und 20 Pollen nachfrägt. Unabhängig davon, ob die Blüte eine bereits bestehenden Pflanze erweitert oder durch eine neue Pflanze hervorgebracht wird, kann das Simulationsexperiment in einen Deadlock geraten. Dieser tritt auf, wenn beispielsweise die Blüten f_1 und f_3 ihre angebotenen Pollen jeweils austauschen und Blüte f_2 dabei nicht berücksichtigt wird. Dadurch sind f_1 und f_3 befruchtet und werden von ihren Pflanzen zerstört. Blüte f_2 hingegen hat keine Möglichkeit mehr bestäubt zu werden, da es keine Blüten gibt, die ihre angebotenen Pollen

benötigen oder die von ihr benötigten Pollen anbieten. Diesem ungewollten Verhalten kann nur durch Veränderung des statischen Modells entgegengewirkt werden.

Um die Auswirkungen einer Veränderung der Blütenanzahl messen zu können ohne das statische Modell grundlegend zu verändern, werden dem bisherigen Modell keine einzelnen Blüten, sondern Blütenpaare hinzugefügt. Jedes Blütenpaar besitzt jeweils seine eigene Art, so dass ihre angebotenen und nachgefragten Pollen nicht mit einem anderen Blütenpaar kollidieren. Wird dem einfachen Simulationsmodell ein weiteres Blütenpaar hinzugefügt, so entsteht folgendes Simulationsmodell (siehe Abbildung 5.3) und Simulationsexperiment (siehe Abbildung 5.6) mit vier Blüten.

Gattungen:	g_1	**Ordnungen:**	o_1
Arten:	s_1, s_2	**Regeln:**	$s_1 \rightarrow g_1$
Pflanzen:	p_1, p_2, p_3, p_4		$s_2 \rightarrow g_1$
Blüten:	$f_1 = (g_1, s_1, p_1, 20, 20, 3, 10)$		$o_1 \rightarrow g_1$
	$f_2 = (g_1, s_2, p_2, 20, 20, 3, 10)$	**Bestäuber:**	$v_1 = (o_1, 6, \emptyset, \emptyset)$
	$f_3 = (g_1, s_1, p_3, 20, 20, 3, 10)$		$v_2 = (o_1, 6, \emptyset, \emptyset)$
	$f_4 = (g_1, s_2, p_4, 20, 20, 3, 10)$		$v_3 = (o_1, 6, \emptyset, \emptyset)$
			$v_4 = (o_1, 6, \emptyset, \emptyset)$

Tabelle 5.3: Einfaches Simulationsmodell mit 4 Blüten.

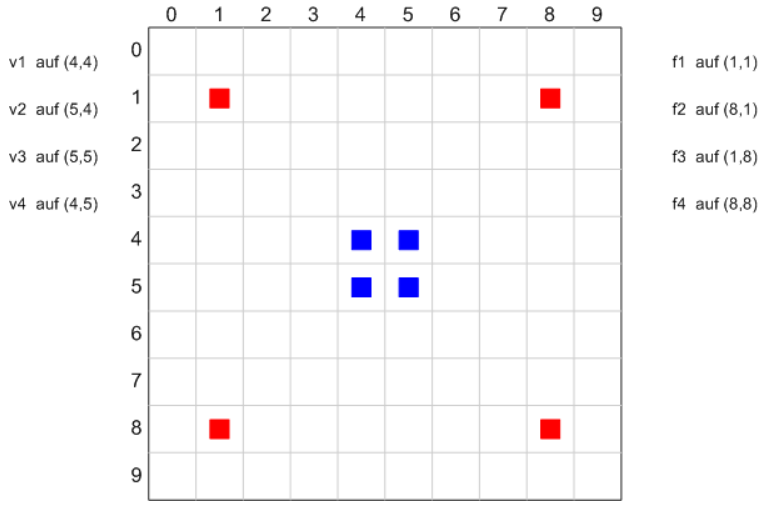

Abbildung 5.6: Einfaches Simulationsexperiment mit 4 Blüten.

Unter der Voraussetzung, dass sich bei gleichbleibender Anzahl an Bestäubern die Anzahl der Blütenpaare erhöht, kann die Skalierbarkeit des Systems im Bezug auf die Blüten anhand der oben aufgestellten Kostendefinition festgestellt werden.
Wie in Abbildung 5.7 zu sehen ist, nehmen die Kosten für jedes Blütenpaar proportional zu, d.h. das System ist im Bezug auf zusätzlich Blüten schlecht skaliert.

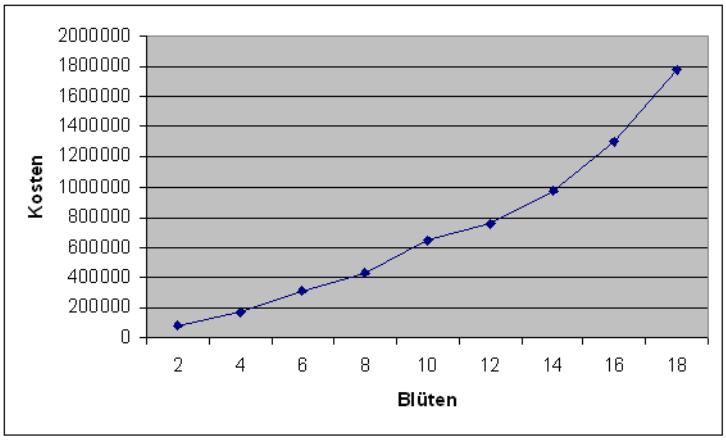

Abbildung 5.7: Kosten eines APS bei steigender Anzahl an Blüten.

5.4 Robustheit und Flexibilität

Wie in den Abschnitten 3.1.2 und 3.1.3 beschrieben, grenzen sich Robustheit und Flexibilität eines Systems dahingehend voneinander ab, dass durch die Robustheit gemessen wird, ob sich ein System trotz unvorhersehbaren, ungünstigen Bedingungen korrekt verhält. Die Flexibilität hingegen misst die Anpassungsfähigkeit eines Systems bei einer dynamischen Veränderung des Modells.

Robustheit: Ein System ist laut Abschnitt 3.1.2 robust, wenn es trotz eines ungewollten Verhaltens einzelner Komponenten immer noch zuverlässig funktioniert. Im Folgenden werden die Möglichkeiten des ungewollten Verhaltens aufgelistet und die Robustheit des Systems bei deren Auftreten bewertet:

Eine Blüte verhält sich falsch: Bei der Produktion der falschen Anzahl an Pollen kann das System nicht zuverlässig funktionieren, da die Anzahl der angebotenen wie auch die Anzahl der nachgefragten Pollen in einem System immer aufeinander abgestimmt sind. Sollte demnach eine Blüte beispielsweise weniger Pollen anbieten als ursprünglich vorgesehen, so wird die Blüte, die diese Pollen benötigt nicht vollständig bestäubt bzw. befruchtet. Alle anderen Systemkomponenten funktionieren unabhängig von diesem Fehlverhalten weiterhin korrekt. Sollte eine Blüte hingegen zu viele Pollen produzieren, so können dadurch die Bestäuber des Systems ausfallen. Sollte beispielsweise eine Blüte 10 Pollen zu viel anbieten und ein Bestäuber bei einer Kapazität von 6 genau 6 dieser 10 Pollen geladen haben, so hat der Bestäuber keine Möglichkeit die Pollen abzugeben und fällt demnach für den Transport anderer Pollen aus. Sollte dieser Bestäuber der einzige des Systems sein, so bricht das ganze System zusammen, da keine Pollen mehr transportiert werden können. Eine erhöhte Pollenproduktion hat demnach eine negative Veränderung des ganzen Systems zur Folge.

Eine Pflanze verhält sich falsch: Wie eben beschrieben, ist die Anzahl der angebotenen und nachgefragten Pollen der einzelnen Blüten aufeinander abgestimmt. Sollte eine Pflanze jedoch zu wenige oder falsche Blüten produzieren, so wird dieses Gleichgewicht

erheblich gestört. Welche Auswirkungen ein falsches Verhalten einer Pflanze hat, kommt auf die angebotenen bzw. nachgefragten Pollen der falsch produzierten Blüten an. Demnach liegt das Augenmerk weniger auf der Anzahl der falsch oder nicht produzierten Blüten, sondern vielmehr auf dem Ausfall bzw. das Hinzukommen neuer Pollen.

Eine Pflanze bzw. Blüte fällt vollständig aus: Auch der Ausfall einer vollständigen Pflanze bzw. Blüte stört das interne Pollengleichgewicht. Ein Ausfall einer Blüte entspricht einem Wegfall sämtlicher angebotenen wie auch nachgefragten Pollen. Sollte beispielsweise eine Blüte ausfallen, die Pollen angeboten hat, so wird durch deren Ausfall nur die Blüte beeinflusst, die diese Pollen benötigt. Die restlichen Komponenten des Systems funktionieren weiterhin fehlerfrei. Sollte jedoch eine Blüte ausfallen, die Pollen benötigt hat, so hat dies ein negatives Verhalten des Systems bis hin zu dessen Zusammenbruch zur Folge, da die für die ausgefallene Blüte produzierten Pollen einen Bestäuber blockieren können.

Ein Bestäuber verhält sich falsch: Sollte ein Bestäuber mehr oder weniger Pollen tragen, als ursprünglich vorgesehen, so hat dies zur Folge, dass sich die Effektivität des Systems leicht ändert, weil er eventuell öfter bzw. weniger oft von einer Blüte zu einer anderen Blüte fliegen muss. Das System jedoch funktioniert neben einer Veränderung der Effektivität weiterhin zuverlässig.

Sollte ein Bestäuber falsche Pollen transportieren, so kommt es stark darauf an, ob es weitere Bestäuber gibt, die sich um den Transport der ursprünglich vorgesehenen Pollen kümmern. Sollte dies der Fall sein, so müssen nur leichte Änderungen der Effektivität in Kauf genommen werden. Sollte es jedoch keine anderen Bestäuber für die ursprünglich vorgesehene Pollenart geben, so wird sich das System falsch verhalten, da die Blüten, die diese Pollen anbieten bzw. nachfragen nie bestäubt werden.

Ein Bestäuber fällt vollständig aus: Der Ausfall eines Bestäubers kann abhängig von seinem Zustand verschieden schwere Auswirkungen auf das System haben. Sollte eine System nur einen Bestäuber haben, so bricht es mit dessen Ausfall vollständig zusammen. Fällt ein Bestäuber in einem System mit mehr als einem Bestäuber aus, so hat dies nur leichte Veränderungen der Effektivität zur Folge, falls der Bestäuber unbeladen war. Fällt jedoch ein beladener Bestäuber aus, so kann die Blüte, die diese Pollen benötigt, nicht vollständig bestäubt werden. Dieser Ausfall hat jedoch neben einer leichten Veränderung der Effektivität keine Auswirkungen auf die anderen Komponenten des Systems.

Das System der künstlichen Blütenbestäubung verhält sich nach diesen Erkenntnissen teilweise robust. Sollte ein falsches Verhalten oder ein Ausfall einer Pflanze, einer Blüte oder eines Bestäubers eine größere Menge an angebotenen als nachgefragten Pollen zur Folge haben, so hat das System mit großer Wahrscheinlichkeit einen hohen Effektivitätsverlust oder bricht sogar zusammen. Sollte ein fehlerhaftes Verhalten bzw. ein Ausfall eine größere Menge an nachgefragten als angebotenen Pollen zur Folge haben, so läuft das System mit hoher Wahrscheinlichkeit fehlerfrei weiter und nur auf die von dem Verlust der Pollen betroffene Pflanze wirken sich die Veränderungen aus.

Flexibilität: Ein System ist laut Abschnitt 3.1.3 flexibel, wenn es sich dynamisch an Veränderungen des Modells anpasst. Diese Veränderungen können ungewollt durch fehlerhafte Sys-

temkomponenten oder aber gewollt entstehen. Die Erzeugung einer neuen Blüte während der Laufzeit kann demnach wie im biologischen Vorbild der künstlichen Blütenbestäubung eine durchaus gewollte Veränderung des Modells darstellen. Im Folgenden wird die Flexibilität der künstlichen Blütenbestäubung anhand möglicher Veränderungen des Modells diskutiert.

Ein oder mehrere Bestäuber fallen aus bzw. werden hinzugefügt: Durch einen Ausfall eines Bestäubers, erhöht sich die Arbeit der restlichen Bestäuber. Die zu dem Zeitpunkt des Ausfalls von Blüten angebotenen Pollen werden nun von weniger Bestäubern auf ihre empfangenden Blüten transportiert, d.h. der Ausfall eines Bestäubers führt dazu, dass die restlichen Bestäuber dessen Arbeit übernehmen.

Da sich die Bestäuber einzig nach den Duftstoffen der Blüten richten, passt sich ein neu eingefügter Bestäuber automatisch an und übernimmt mit dem Transport von Pollen einen Teil der Arbeit der anderen Bestäuber. Ein neuer Bestäuber passt sich demnach auf Basis der gerochenen Duftstoffe flexibel dem aktuellen Zustand des Systems an.

Ein oder mehrere Blüten fallen aus bzw. werden hinzugefügt: Auf den Ausfall einer Blüte reagiert das System flexibel, aber zeitlich leicht versetzt. Hat eine Blüte kurz vor ihrem Ausfall Duftstoffe abgegeben, so ist es möglich, dass sich ein Bestäuber von der nicht mehr existenten Blüte angezogen fühlt und umsonst in deren Richtung fliegt. Sobald die zuletzt abgegebenen Duftstoffe aus der Umgebung verschwunden sind, verhält sich das System, als wäre die Blüte nie da gewesen. Demnach passt es sich flexibel an die Veränderung an.

Wird eine neue Blüte in das Modell eingefügt, so reagieren die Bestäuber des Systems auf diese Blüte, sobald sie Duftstoffe abgibt. Durch das Anlocken von Bestäubern durch ihre Duftstoffe gliedert sie sich in das System ein. Die Bestäuber und damit das System reagieren auf Basis der abgegebenen Duftstoffe flexibel auf einen neue Blüte.

Ein oder mehrere Pflanzen fallen aus bzw. werden hinzugefügt: Der Ausfall einer Pflanze ohne Blüte wird von dem System ignoriert, da eine Pflanze nicht relevant für eine Bestäubung ist. Bei einem Ausfall einer Pflanze mit Blüte reagiert das System gleich flexibel wie bei einem Ausfall einer Blüte ohne deren Pflanze.

Das Hinzufügen einer neuen Pflanze ohne Blüten wird von dem System ebenfalls ignoriert, da eine weitere Pflanze für eine Bestäubung der existierenden Blüten nicht relevant ist. Wird hingegen eine Pflanze mit einer Blüte hinzugefügt, so verhält sich das System gleich flexibel wie bei einem Hinzufügen einer Blüte ohne Pflanze.

Abschließend kann gesagt werden, dass sich das System bei Ausfall oder Hinzukommen einer neuen Komponente immer flexibel an die neue Situation anpasst.

5.5 Ergänzende Simulationsergebnisse

Die in den vorangegangenen Abschnitten errechneten Ergebnisse basieren auf den in Abschnitt 3.1 angegebenen Evaluationskriterien. Weitere erwähnenswerte Simulationsergebnisse werden in diesem Abschnitt gesammelt dem Leser vorgestellt.

5.5.1 Effizienz

Wie in Abschnitt 3.1.4 beschrieben, kann die Effizienz des Systems nur an dessen Dauer gemessen werden. Ein Vergleich mit realen Werten ist nicht möglich, da das formale Modell bisher in keiner realen Umgebung eingesetzt wird. Wie Abbildung 5.8 des einfachen Simulationsmodells zu entnehmen ist, nimmt die durchschnittliche Dauer eines Simulationsexperiments mit der Zunahme der Bestäuber ab und konvergiert gegen einen bestimmten Wert. Dieses Verhalten, d.h. die Annäherung an eine bestimmte Dauer, lässt sich in jedem Simulationsexperiment beobachten.

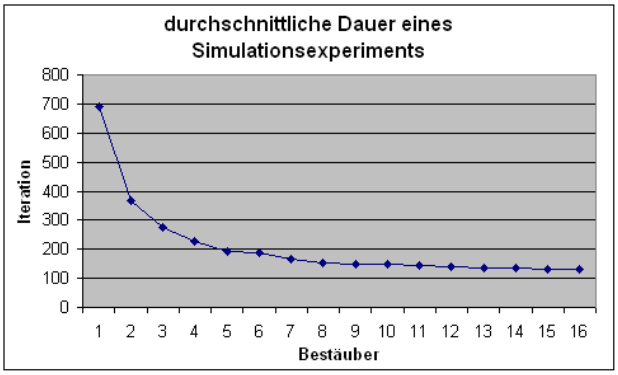

Abbildung 5.8: Annäherung der Dauer eines Simulationsexperiments an einen bestimmten Wert.

5.5.2 Parametereinstellungen

Die bisherigen Ergebnisse wurden unter festen Parametereinstellungen ermittelt. Dieser Abschnitt soll eine Einschätzung liefern, welche Auswirkungen das Verstellen einzelner Parameter für das Modell haben kann. Für einen Überblick der veränderbaren Parameter sei hier auf Anhang A verwiesen. Im Folgenden werden die Veränderungen einiger Parameter und ihre Auswirkungen genauer beleuchtet:

Intensität eines Duftstoffes: Die Intensität eines Duftstoffes bietet einerseits eine Bewertungsgrundlage für die Aktualität eines Duftstoffes, gibt jedoch andererseits auch an, wie weit ein Duftstoff von seiner Quelle weg propagiert werden kann. Demnach kann durch die Intensität der Einzugsbereich einer Blüte gesteuert werden, d.h. ist die Intensität eines Duftstoffes hoch, so können damit auch Bestäuber in einer größeren Entfernung angelockt werden. Durch eine Simulation zum Vergleich der Intensitäten wird deutlich, dass sich die Dauer eines Simulationsexperiments mit zunehmender Intensität an einen festen Wert annähernd verringert (siehe Abbildung 5.4). Um die optimale Intensität eines Modells zu erlangen, muss jedoch neben einer Verbesserung der Simulationsdauer der zusätzliche Aufwand für die propagierten Duftstoffe berücksichtigt werden.

Wie in Abbildung 5.5 zu sehen ist, steigt die Effektivität der Simulationsexperimente mit zunehmender Intensität an, stagniert und nimmt annähernd an einen bestimmten Wert wieder ab. Dieser Verlauf basiert darauf, dass die Bestäuber durch eine erhöhte Intensität der

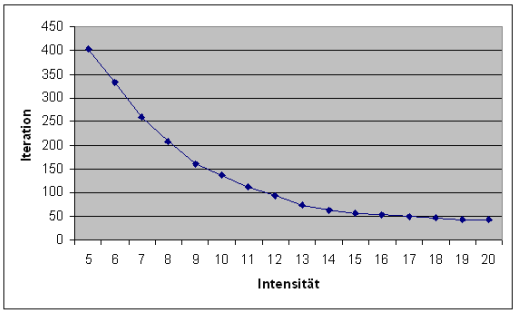

Tabelle 5.4: Abnehmende Dauer bei steigender Intensität.

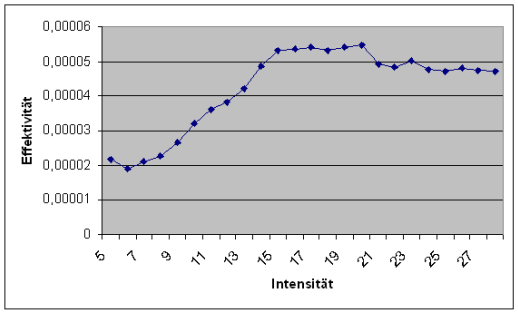

Tabelle 5.5: Verlauf der Effektivität in Abhängigkeit von einer steigenden Intensität.

Duftstoffe schneller zu einer Blüte finden. Wird jedoch das ganze Gridfeld von den Duftstoffen geflutet, so verbessert sich die Schnelligkeit der Bestäuber zum Auffinden einer Blüte nicht mehr. Demnach erhöht sich die Effektivität ab einer speziellen Intensität ebenfalls nicht mehr. Durch die weitere Erhöhung der Intensität steigt jedoch die Anzahl der Duftstoffe pro Iteration auf dem Gridfeld bis zu einem maximalen Wert (hier 10*10*2 = 200) und stagniert bei diesem Wert. Demnach fällt die Effektivitätskurve bis sie bei einen bestimmten Wert stagniert.

Erhöhung der Intensität eines Duftstoffes: Durch diesen Parameter kann eine dynamische Erhöhung der Intensität eines Duftstoffes während der Simulation festgelegt werden. Demnach wird die Intensität um einen festen Wert erhöht, so lang wie die Blüte von keinem Bestäuber angeflogen wird. Nach einer Bestäubung wird die Intensität zurückgesetzt und erneut regelmäßig erhöht. Durch eine dynamische Erhöhung der Intensität des Duftstoffes verbessert sich die Simulationsdauer (wie in Abbildung 5.6 zu sehen ist), jedoch muss die Erhöhung der Duftstoffe bzw. Nachrichten ebenfalls berücksichtigt werden.

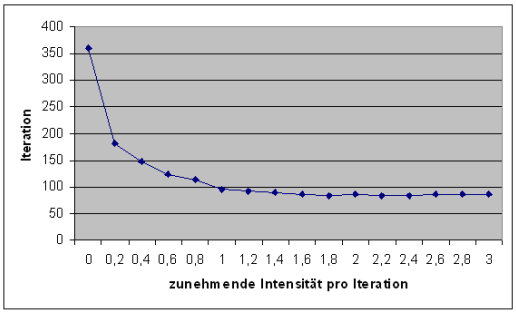

Tabelle 5.6: Abnehmende Dauer bei steigender Intensitätserhöhung.

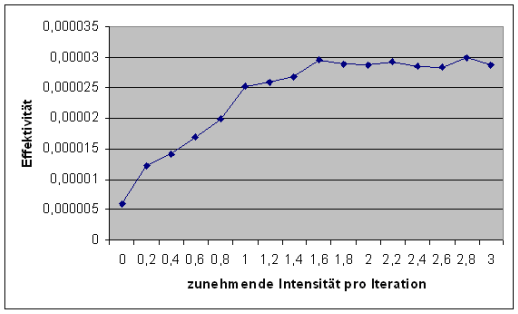

Tabelle 5.7: Verlauf der Effektivität in Abhängigkeit von einer steigenden Intensitätserhöhung.

Wie in Abbildung 5.7 zu sehen ist, steigt die Effektivität mit zunehmender Intensitäts-

erhöhung an. Dieses Verhalten resultiert daraus, dass bei steigender Intensitätserhöhung die Duftstoffe weiter von ihrer Blüte weg propagiert werden und somit die Wahrscheinlichkeit erhöhen, dass ein Bestäuber durch sie angelockt wird. Dadurch finden die Bestäuber bei steigender Intensitätserhöhung die für sie interessanten Blüten schneller und können diese demnach schneller bestäuben. Durch dieses Verhalten erhöht sich die Effektivität des Simulationsexperiments. Die zusätzlichen Kosten für die Propagation der Duftstoffe sind zu dem entstandenen Nutzen vergleichsweise klein und fallen aus diesem Grund bei der Berechnung der Effektivität weniger ins Gewicht.

Verringerung der Intensität eines Duftstoffes innerhalb der Umgebung: Die Verringerung der Intensität innerhalb der Umgebung steht in direktem Kontrast zu der Höhe der Intensität. Soll der Blüte ein größerer Einzugsbereich gewährt werden, so kann einerseits die Intensität der Duftstoffe erhöht, andererseits auch der Parameter für die Verringerung der Duftstoffe verändert werden. Dadurch wird festgelegt, wie stark ein Duftstoff auf einem Feld an Intensität abnimmt. Je stärker die Abnahme, desto weniger weit wird der Duftstoff propagiert.

Frequenz der ausgesendeten Duftstoffe: Je öfter eine Blüte Duftstoffe aussendet, umso größer sind die Erfolgsaussichten auf eine Bestäubung. Durch ihre Duftstoffe zeigt die Blüte eine Präsenz in ihrem Umfeld, d.h. kommt ein Bestäuber in ihr Umfeld, so findet er bei einer starken Frequenz mit hoher Wahrscheinlichkeit einen aktuellen Duftstoff der Blüte vor. Sendet eine Blüte jedoch mit sehr geringer Frequenz, beispielsweise alle 30 Iterationen, so ist die Wahrscheinlichkeit sehr hoch, dass ein Bestäuber im Umfeld der Blüte keinen bzw. einen alten Duftstoff vorfindet und sich auf Grund der Informationen für einen Duftstoff einer konkurrierenden Blüte entscheidet. Die Frequenz des Duftstoffes einer Blüte ist demnach, neben der Intensität und der Konzentration, entscheidend für die Anziehung eines Bestäubers. Es muss jedoch beachtet werden, dass durch eine hohe Frequenz an Duftstoffen ein Mehraufwand entsteht, da die Blüte eine Vielzahl an Duftstoffen produzieren und das Netzwerk deren Weiterverbreitung durchführen muss.

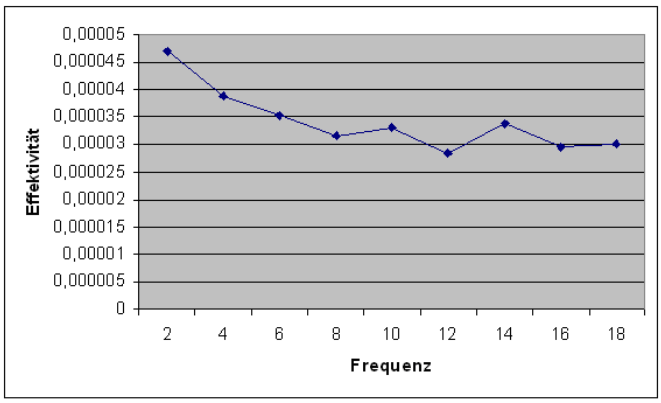

Tabelle 5.8: Verlauf der Effektivität in Abhängigkeit von einer steigenden Sendefrequenz der Duftstoffe.

Wie in Abbildung 5.8 zu sehen ist, arbeitet das System mit einer möglichst hohen Frequenz am effektivsten. Die Kosten, die durch die zusätzlichen Duftstoffe entstehen, sind zu dem daraus gezogenen Nutzen vergleichsweise klein. Bei einem realen Einsatz der künstlichen Blütenbestäubung muss jedoch beachtet werden, dass das Netzwerk durch die erhöhten Signale bzw. Duftstoffe nicht überlastet wird.

Konzentration der Belohnungen: Bestäuber werden in der Biologie in erster Linie durch die Konzentration der Belohnungen einer Blüte angelockt. In dem Modell der künstlichen Blütenbestäubung gilt die Konzentration ebenfalls als Lockmittel. Hat ein Bestäuber die Auswahl zwischen zwei Duftstoffen, deren relevante Werte bis auf die Konzentration identisch sind, so folgt er dem Duftstoff mit höherer Konzentration. Durch diesen Wert kann eine Blüte ihre Anziehungskraft verändern. Jedoch schlägt eine veränderte Konzentration nur dann zu Buche, wenn sich die ausgesandten Duftstoffe der Blüten auf den gleichen Feldern befinden, d.h. erst bei einer Auswertung verschiedener Duftstoffe durch den Bestäuber, hat eine höhere Konzentration relevante Auswirkungen.

Kapazität eines Bestäubers: Durch die Kapazität des Bestäubers kann die durchschnittliche Simulationsdauer verändert werden. Bietet eine Blüte beispielsweise 20 Pollen an, so wird ein Bestäuber mit der Kapazität von maximal 20 Pollen das optimalste Ergebnis erreichen, da er die Blüte nur einmal anfliegen muss, um alle Pollen aufzunehmen. Jeder Bestäuber mit einer höheren Kapazität wird einerseits in voraussichtlich der gleichen Zeit fertig, kann jedoch seine Kapazität nicht voll auslasten. Dadurch wird die Effektivität des Systems verringert. Ein Bestäuber mit geringerer Kapazität hat zwar eine bessere Auslastung, benötigt jedoch mehr Zeit für den Transport von 20 Pollen. Demnach ist die Wahl einer passenden Kapazität stark mit der Effektivität des Systems verbunden.

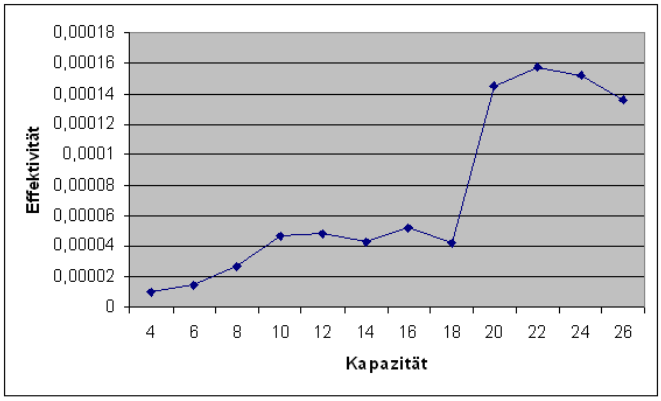

Tabelle 5.9: Verlauf der Effektivität in Abhängigkeit von einer steigenden Kapazität.

Wie in Abbildung 5.9 zu sehen ist, steigt die Effektivität des Simulationsexperiments mit der Erhöhung der Kapazität. Die Sprünge der Kurve ergeben sich durch den Zusammenhang zwischen der Kapazität eines Bestäubers und den angebotenen Pollen. Eine Blüte in dem einfachen Simulationsmodell bietet 20 Pollen an, d.h. eine Kapazität, dessen Vielfaches der

Anzahl der angebotenen Pollenmenge entspricht, ist für die Bestäuber am geeignetsten. Es fällt jedoch auf, dass die Simulationsexperimente mit einer Kapazität von 21 durchschnittlich besser ausfallen als die Experimente mit einer Kapazität von 20. Dieses Ergebnis ist verwunderlich, da ein Bestäuber immer maximal 20 Pollen, d.h. die Menge der angebotenen Pollen einer Blüte, transportieren wird. Die Bestäuber mit einer Kapazität von 21 erzeugen auf Grund der unausgelasteten Kapazitäten höhere Kosten und müssten auf Grund dessen schlechter abschneiden als die Bestäuber mit Kapazität 20. Die leichten Abweichungen der Ergebnisse liegen voraussichtlich in der Menge der durchgeführten Simulationsexperimente. Diese Vermutung konnte jedoch aus zeitlichen Gründen im Rahmen dieser Arbeit weder bestätigt noch widerlegt werden.

5.6 Fallbeispiel: Fahrerlose Transportfahrzeuge im Bereich der Krankenpflege

In den bisherigen Abschnitten wurde das Modell der künstlichen Blütenbestäubung allgemein evaluiert. Um zu zeigen, dass das formale Modell nicht nur ein theoretischer, sondern auch ein in der Praxis einsetzbarer Ansatz ist, widmet sich dieses Kapitel einem konkreten Szenario im Bereich der Krankenpflege. Nach dessen Vorstellung in Abschnitt 5.6.1 werden die relevante Komponenten des Szenarios in Abschnitt 5.6.2 auf das formale Modell der künstlichen Blütenbestäubung abgebildet. Auf Basis dieser Abbildung kann in Abschnitt 5.6.3 ein konkretes Simulationsmodell entwickelt werden, dessen Simulation eine Bewertung der künstlichen Blütenbestäubung im Bereich der Krankenpflege zulässt.

5.6.1 Szenario

Um die Wirkungsweise des formalen Modells in einer realen Umgebung zu verdeutlichen, wird ein Szenario im Bereich der Krankenpflege gewählt.

Ein Krankenhaus besteht aus einer Vielzahl an Patienten, die während ihres Krankenhausaufenthaltes jeweils in einem Raum untergebracht sind und ihr eigenes Bett haben. Regelmäßig wird den Patienten u.a. Essen gebracht und die Bettwäsche gewechselt. Um diese Aufgaben erledigen zu können, muss das Krankenhauspersonal regelmäßig frische Bettwäsche holen und die alte Bettwäsche in einem Sammelbehälter abtransportieren. Das Essen muss ebenfalls aus der Kantine auf die einzelnen Stationen gebracht und nach Verzehr wieder abtransportiert werden. Um diese “Hintergrundarbeiten” zu erleichtern und keine teuren Arbeitskräfte des Krankenhauspersonals damit zu belasten, kann beispielsweise der Einsatz eines fahrerlosen Transportsystems in Betracht gezogen werden. Demnach würden fahrerlose Transportfahrzeuge auf vorgefertigten Bahnen innerhalb einer Station umherfahren und beispielsweise frische Wäsche ausgeben bzw. schmutzige Wäsche einsammeln.

Die Koordination der fahrerlosen Transportfahrzeuge kann dezentral mit Hilfe des Modells der künstlichen Blütenbestäubung erfolgen. Dies setzt jedoch voraus, dass die relevanten Bestandteile einer Krankenhausstation identifiziert und auf die Komponenten des formalen Modells abgebildet werden.

5.6.2 Abbildung des Szenarios auf das künstliche Modell

Die für den Einsatz von fahrerlosen Transportfahrzeugen relevanten Bestandteile eines Krankenhauses sind

- das Patientenbett inklusive einem Tisch für das Essen und einem Wäschekorb für schmutzige Wäsche
- die zentrale Wäscheannahme, d.h. ein großer zentraler Wäschekorb
- die zentrale Essensausgabe

Diese Bestandteile werden auf das formale Modell der künstlichen Blütenbestäubung abgebildet. Wie in Abbildung 5.9 zu sehen ist, entsprechen das Patientenbett, der zentrale Wäschekorb wie auch die zentrale Essensausgabe einer Pflanze des künstlichen Modells.

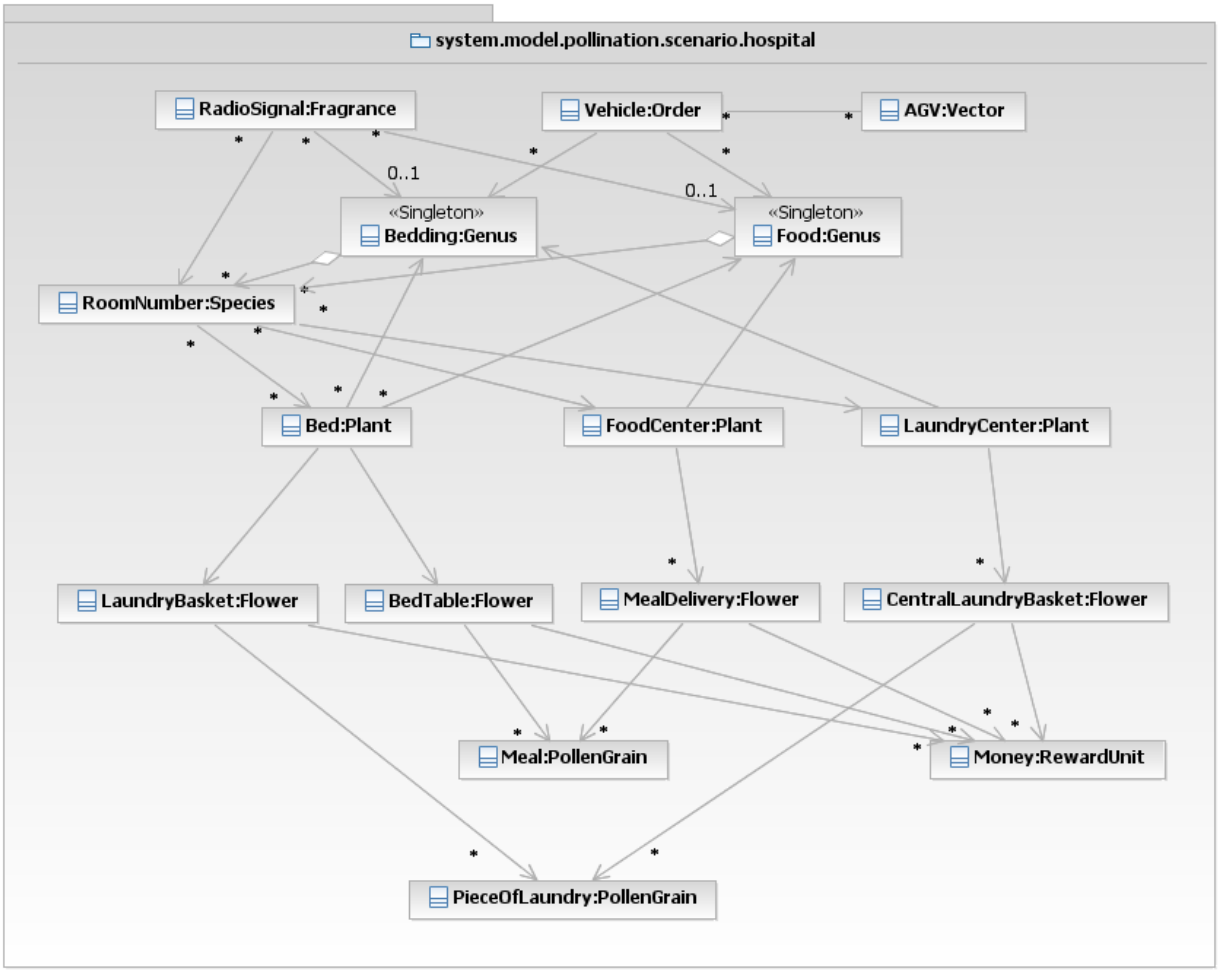

Abbildung 5.9: Abbildung des Krankenhausszenarios auf das künstliche Modell.

Das Bett besitzt zwei Blüten, den lokalen Wäschekorb und den Tisch, auf dem der Patient das Essen stellen kann. Der zentrale Wäschekorb besitzt mehrere Wäscheannahmestellen, die die schmutzige Bettwäsche jeweils eines Bettes nachfragen. Auch die Essensausgabe besitzt mehrere Ausgabestellen, die das Essen für jeweils einen Patienten zur Verfügung stellen. Diese Umsetzung ist dahingehend sinnvoll, da es unter den Mahlzeiten der Patienten Unterschiede geben kann. So muss ein Patient, der nur fettarme Mahlzeiten zu sich nehmen darf, diese auch geliefert bekommen. Ein fahrerloses Transportfahrzeug (AGV = Automated Guided Vehicle) wird auf die Komponente Bestäuber abgebildet und transportiert, abhängig davon welcher Ordnung es angehört, Bettwäsche oder Essen. Für jedes abgeholte bzw. gelieferte Essen oder Stück Bettwäsche erhält das Transportfahrzeug eine Belohnungseinheit.

5.6.3 Simulation und Ergebnisauswertung

Für die Simulation des Krankenhausszenarios wird ein konkretes Simulationsmodell und dessen Eingliederung in eine konkrete Umwelt benötigt. Im Folgenden werden die Komponenten des Krankenhauses auf Basis des in Abschnitt 2.3 definierten formalen Modells der künstlichen Blütenbestäubung zu einem Simulationsmodell zusammengefügt:

Gattungen:	g_1, g_2	**Ordnungen:**	o_1, o_2
Arten:	s_1, s_2, s_3, s_4	**Regeln:**	$s_1 \rightarrow g_1$
Pflanzen:	$p_1, p_2, p_3, p_4, p_5, p_6$		$s_1 \rightarrow g_2$
Blüten:	$f_1 = (g_1, s_1, p_1, 0, 4, 3, 10)$		$s_2 \rightarrow g_1$
	$f_2 = (g_1, s_2, p_1, 0, 3, 3, 10)$		$s_2 \rightarrow g_2$
	$f_3 = (g_1, s_3, p_1, 0, 4, 3, 10)$		$s_3 \rightarrow g_1$
	$f_4 = (g_1, s_4, p_1, 0, 2, 3, 10)$		$s_3 \rightarrow g_2$
	$f_5 = (g_2, s_1, p_2, 10, 0, 3, 10)$		$s_4 \rightarrow g_1$
	$f_6 = (g_2, s_2, p_2, 2, 0, 3, 10)$		$s_4 \rightarrow g_2$
	$f_7 = (g_2, s_3, p_2, 5, 0, 3, 10)$		$o_1 \rightarrow g_1$
	$f_8 = (g_2, s_4, p_2, 7, 0, 3, 10)$		$o_2 \rightarrow g_2$
	$f_9 = (g_1, s_1, p_3, 4, 0, 3, 10)$	**Bestäuber:**	$v_1 = (o_1, 5, \emptyset, \emptyset)$
	$f_{10} = (g_2, s_1, p_3, 0, 10, 3, 10)$		$v_2 = (o_1, 5, \emptyset, \emptyset)$
	$f_{11} = (g_1, s_2, p_4, 3, 0, 3, 10)$		$v_3 = (o_2, 20, \emptyset, \emptyset)$
	$f_{12} = (g_2, s_2, p_4, 0, 2, 3, 10)$		$v_4 = (o_2, 20, \emptyset, \emptyset)$
	$f_{13} = (g_1, s_3, p_5, 4, 0, 3, 10)$		
	$f_{14} = (g_2, s_3, p_5, 0, 5, 3, 10)$		
	$f_{15} = (g_1, s_4, p_6, 2, 0, 3, 10)$		
	$f_{16} = (g_2, s_4, p_6, 0, 7, 3, 10)$		

Tabelle 5.10: Simulationsmodell des Krankenhausszenarios.

In dem konkreten Modell 5.10 entspricht Pflanze p_1 dem zentralen Bettwäschekorb, Pflanze p_2 hingegen der zentralen Essensausgabe. Die Pflanze p_3, p_4, p_5 und p_6 repräsentieren die Patientenbetten der Krankenhausstation. Die beiden zentralen Pflanzen besitzen jeweils vier Blüten, die wiederum Pollen für jeweils ein Bett nachfragen oder anbieten. Jedes Bett besitzt zwei Blüten, den Tisch und den Wäschekorb, die ebenfalls Pollen von der Essensausgabe nachfragen oder für den zentralen Wäschekorb anbieten. Für eine Belieferung der Patienten mit Essen sind zwei fahrerlose Transportfahrzeuge zuständig. Das Abholen der schmutzigen Wäsche wird von zwei anderen fahrerlosen

Transportfahrzeugen erledigt.

Um das Modell in eine konkrete Umwelt einzubetten, wurde ein 10x10 großes Gridfeld verwendet. Wie in Abbildung 5.10 zu sehen ist, stehen die vier Betten in möglichst ähnlichen Abständen zu den zentralen Lagern, um eine Bevorzugung eines bestimmten Bettes so gut wie möglich zu vermeiden.

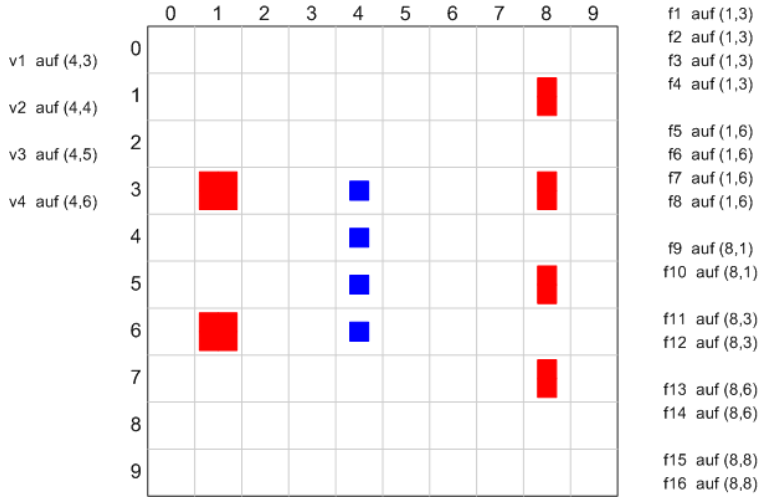

Abbildung 5.10: Simulationsexperiment des Krankenhausszenarios.

Durch eine Simulation des eben beschriebenen Krankenhausszenarios können u.a. Aussagen über die Skalierbarkeit, die Robustheit wie auch die Flexibilität des Systems getroffen werden.

Skalierbarkeit

Um festzustellen, ob ein System skalierbar ist, werden die Kosten für zusätzliche Komponenten des Systems ermittelt und bewertet. Da die Anzahl der Patienten auf einer Krankenhausstation nie gleich bleibt, ist die Skalierbarkeit des Systems in Bezug auf die Anzahl der Betten von Interesse. Des Weiteren sollte auch die Steigung der Kosten für jedes weitere fahrerlose Transportfahrzeug innerhalb des Szenarios betrachtet werden.

Wie in Abbildung 5.11 zu erkennen ist, steigen die Kosten für jedes weitere Bett proportional an. Das System ist demnach in Bezug auf die Anzahl der Patientenbetten schlecht skaliert. In Abbildung 5.12 hingegen ist trotz der etwas “wackeligen” Kurve zu erkennen, dass die Kosten für jedes weitere Transportfahrzeug linear ansteigen. Dies deckt sich mit den Ergebnissen der Skalierbarkeit des einfachen Modells aus Abschnitt 5.3.

Neben einer Skalierbarkeit des Systems ist auch die optimale Anzahl an fahrerlosen Transportfahrzeugen für die vorhandene Menge an Betten interessant. Bei der Berechnung muss jedoch berücksichtigt werden, dass das Krankenhausmodell zwei Ordnungen besitzt, deren Bestäuber jeweils verschiedene Gattungen anfliegen. Bei der Erhöhung der Bestäuberanzahl darf demnach nicht nur ein Bestäuber hinzugefügt werden. Es müssen immer mindestens zwei, d.h. für jede Ordnung ein, Bestäuber angelegt werden. Bei der Simulation dieses Simulationsmodells genügt es pro Simulationsexpriment genau zwei zusätzliche Bestäuber hinzuzufügen, da jede Blüte durch das Anfliegen

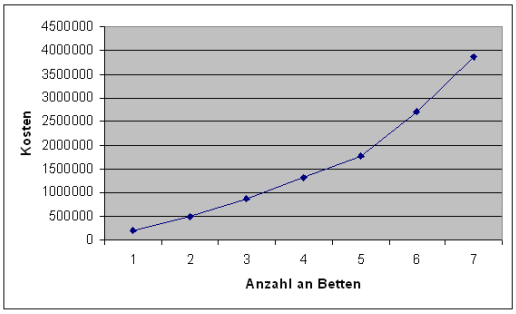

Tabelle 5.11: Kosten des Krankenhausmodells bei steigender Anzahl an Betten.

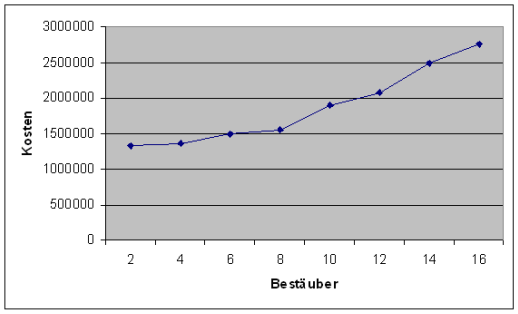

Tabelle 5.12: Kosten des Krankenhausmodells bei steigender Anzahl an fahrerlosen Transportfahrzeugen.

eines Bestäubers bereits befruchtet ist. Die Anzahl der angebotenen bzw. nachgefragten Pollen jeder Blüte ist kleiner als die maximale Kapazität eines passenden Bestäubers.

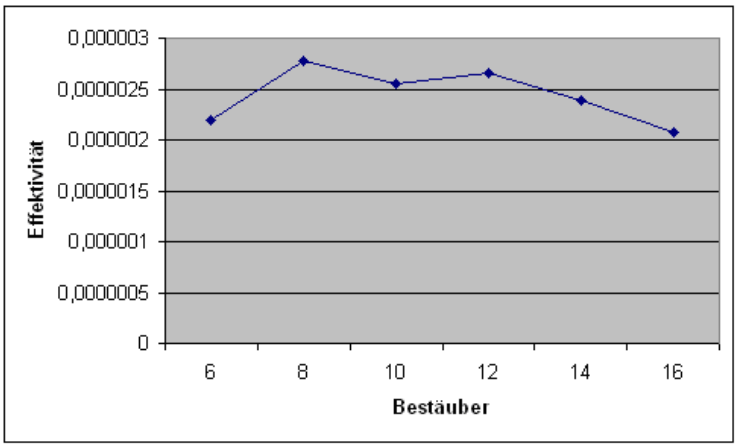

Abbildung 5.11: Effektivität der Krankenhausszenarios bei steigender Bestäuberanzahl.

Wie in Abbildung 5.11 zu sehen ist, verhält sich das Simulationsmodell bei einer Bestäuberanzahl von 8, d.h. 4 der Ordnung o_1 und 4 der Ordnung o_2, am effektivsten.

Robustheit

Die Robustheit des Systems der künstlichen Blütenbestäubung ist im Bereich der Krankenpflege besonders relevant.

So entspricht eine Entlassung bzw. eine Verlegung eines Patienten auf eine andere Station dem Ausfall einer kompletten Pflanze. Sollte dies der Fall sein, müssen die zentrale Essensausgabe wie auch der zentrale Wäschekorb über die Veränderung benachrichtigt werden, da sonst ein fahrerloses Transportfahrzeug das Essen des entlassenen Patienten auf der Station umherfährt, ohne es abgeben zu können. Sollte hingegen ein Patient neu auf die Station kommen, so müssen die zentra-

len Ausgabe- bzw. Annahmestellen ebenfalls darüber benachrichtigt werden, da der Patient sonst beispielsweise kein Essen erhält.

Sollte hingegen ein Transportfahrzeug ausfallen, welches eine Mahlzeit transportiert hat, muss die Essensausgabe dem Patienten erneut eine Mahlzeit zur Verfügung stellen. Sollte dies nicht der Fall sein, bekommt der betroffene Patient kein Essen.

Der Ausfall einer Blüte ist in dem Krankenhausmodell nicht direkt möglich, da eine Blüte durch ein physisches Objekt z.B. einen Tisch repräsentiert wird und somit nicht ausfallen kann. Jedoch kann es passieren, dass die Blüte keine Duftstoffe mehr sendet. Dadurch werden die fahrerlosen Transportfahrzeuge nicht auf die Bedürfnisse der Blüte bzw. des Patienten aufmerksam und bedienen sie demnach nicht. Dies führt dazu, dass ein Patient weder Essen bekommt noch dessen alte Bettwäsche abtransportiert wird.

Flexibilität

Die Flexibilität des Systems im Bereich des Krankenhausmodells ist in Bezug auf die ständig wechselnden Patienten relevant. Das System sollte sich demnach an jede Veränderung des Modells anpassen können.

Kommt beispielsweise ein neuer Patient auf die Station, so sendet dessen Tisch wie auch dessen Bettwäschekorb Signale aus, um eine Mahlzeit für den Patienten zu bekommen und die alte Bettwäsche abzugeben. Das Hinzufügen eines neuen Patientenbettes wird flexibel von den Komponenten des System angenommen, da die Blüten des Bettes durch Signale die fahrerlosen Transportfahrzeuge anziehen.

Wird jedoch ein Bett entfernt, weil ein Patient entlassen wurde, so kann es passieren, dass trotzdem ein fahrerloses Transportfahrzeug mit einer Mahlzeit auf dem Weg zu dem nicht mehr belegten Bett unterwegs ist. Begründet ist dies durch alte, noch existente Signale in der Umwelt, die von dem Bett abgegeben wurden als es noch belegt war. Sobald diese Signale vollständig aus der Umwelt verschwunden sind, passt sich das System flexibel an die neue Situation an.

Der Ausfall eines fahrerlosen Transportfahrzeugs wird ebenfalls flexibel von den restlichen Transportfahrzeugen abgefangen, da sie dessen Arbeiten automatisch übernehmen. Da ein Transportfahrzeug jedoch ein physisch existentes Objekt ist, kann es durch seinen Ausfall den Weg für die anderen Transportfahrzeuge blockieren. Unter diesen Umständen ist das manuelle Eingreifen eines Menschen notwendig.

Kapitel 6

Zusammenfassung und Ausblick

Für die Simulation und Evaluation der künstlichen Blütenbestäubung im Rahmen dieser Arbeit wurde dem Leser zu Beginn das biologische Vorbild wie auch das formale Modell der künstlichen Blütenbestäubung vorgestellt. Da für die Simulation des formalen Modells eine geeignete Simulationsumgebung notwendig ist, wurden eine Vielzahl etablierter Simulationsumgebungen im Bereich der Schwarmintelligenz wie auch der Multi-Agenten-Systeme begutachtet und nach den Bedürfnissen des formalen Modells bewertet. Dabei hat sich herausgestellt, dass keine der betrachteten Simulationsumgebungen für das Modell der künstlichen Blütenbestäubung geeignet ist. Auf Basis dieser Erkenntnis wurde anhand des Wasserfallmodells der *Simulator für selbst-organisierende Systeme* entwickelt, der in der Lage ist die Besonderheiten des künstlichen Modells zu simulieren. Mit Hilfe dieser Simulationsumgebung wurde das bisher nur formal existierende Modell der künstlichen Blütenbestäubung nach speziellen Kriterien (u.a. Flexibilität und Robustheit) evaluiert.

Durch eine allgemeine Evaluation des Modells wurde festgestellt, dass es im Bezug auf seine Bestäuber gut skaliert, in Bezug auf seine Blüten jedoch schlecht skaliert ist. Dies führt zu dem Schluss, dass das Hinzufügen eines weiteren Bestäubers in Bezug auf die Kosten weniger problematisch ist, als das Hinzufügen einer Blüte. Bei der Simulation des Modells wurde jedoch vorausgesetzt, dass auch mehr als ein Bestäuber auf einem Feld platziert werden können. Diese Voraussetzung kann für einen realen Einsatz des Systems eventuell nicht mehr gewährleistet sein. Repräsentiert beispielsweise ein fahrerloses Transportfahrzeug einen Bestäuber (siehe Krankenhausszenario 5.6), so besteht das Problem, dass zwei Transportfahrzeuge nicht gemeinsam auf der gleichen Stelle stehen können und auch mit einem speziellen minimalen Abstand aneinander vorbeifahren müssen. Demnach können sich zu viele Transportfahrzeuge untereinander blockieren und dadurch die Effektivität des Modells senken. Der Simulator kann durch die Umsetzung dieses “realen” Problems in einer zukünftigen Version verbessert werden. Eine weitere Verbesserungsmöglichkeit des Simulators besteht in dem Entfernen wie auch dem Hinzufügen einzelner Modellkomponenten zur Laufzeit, da dies im Rahmen dieser Arbeit zeitlich nicht möglich war. Für eine Verbesserung der Bedienbarkeit des Systems kann die graphische Benutzeroberfläche abhängig von den Systemänderungen erweitert bzw. verändert werden. Eine Vereinfachung der Evaluation der gespeicherten Ergebnisse durch ein eigene Applikation ist ebenfalls für den Benutzer vorteilhaft, da sich eine Evaluation zur Zeit nur manuell durchführen lässt.

Sollte das Modell der künstlichen Blütenbestäubung in einem realen Umfeld eingesetzt werden, so ist es unbedingt erforderlich die für die Evaluation entwickelte Bewertungsgrundlage um die "realen" Kosten der Modellkomponenten zu erweitern. Würde das Modell beispielsweise tatsächlich in einem Krankenhaus anhand von fahrerlosen Transportfahrzeugen umgesetzt werden, so muss der Kaufpreis der Transportfahrzeuge wie auch der Preis für die Sensoren, die Signale abgeben, in die Kosten-Rechnung des Systems aufgenommen werden. Besonders im Bezug auf die Skalierbarkeit muss in einem real existierenden System auf die Kostenkontrolle der physischen Ressourcen wie auch auf eine Kontrolle des Ressourcenverbrauchs geachtet werden. Demnach sollte der Einsatz neuer Komponenten den physischen Ressourcenverbrauch nur linear erhöhen und der Software-Ressourcenverbrauch durch die neuen Komponenten nur linear ansteigen [7].

Abschließend kann festgestellt werden, dass die Effektivität des Modells der künstlichen Blütenbestäubung stark von der Konstellation der eingesetzten Pflanzen, Blüten und Bestäuber wie auch den Parametereinstellungen abhängt. Demnach kann ein optimales Blüten-Bestäuber-Paar nur in Zusammenhang mit den für das Modell eingestellten Parametern ermittelt werden. Sollte sich beispielsweise die Intensität eines Modells ändern, so ändert sich voraussichtlich auch das optimale Blüten-Bestäuber-Paar. Eine zukünftige Aufgabenstellung könnte somit die Entwicklung einer Vorgehensweise sein, die die Identifizierung optimaler Parameter für spezifische Modelle ermöglicht.

Abbildungsverzeichnis

Literaturverzeichnis

[1] W.R. Ashby. Principles of self-organizing dynamic systems. *Journal of General Psychology*, 37:125–128, 1947.

[2] Andreas Bertsch. *Blüten - lockende Signale*. Dynamische Biologie 2. Otto Maier Verlag Ravensburg, 1975.

[3] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford, 1999.

[4] Günter Bruns, Peter Mössinger, Daniel Polani, Ralf Schmitt, Rene Spalt, Thomas Uthmann, and Stefan Weber. Xraptor - a simulation environment for continuous virtual multi-agent systems - user manual. http://citeseer.ist.psu.edu/bruns03xraptor.html.

[5] Linnaeus C. *Systema naturae per rega tria naturae, secundum classes, ordines, genera, species, cum characteribus, differntiis, synomysis, locis*. Stockholm, 10th edition, 1758.

[6] C. Camazine, J.L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princetown university press, 2001.

[7] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems*. International computer science series. Addison-Wesley, Harlow, 2005.

[8] Tom De Wolf and Tom Holvoet. Emergence versus self-organisation: different concepts but promising when combined. In *Engineering Self Organising Systems: Methodologies and Applications*, volume 3464 of *Lecture Notes in Computer Science*, pages 1–15. Springer Verlag, 2005.

[9] Holvoet T. De Wolf T., editor. *Emergence and Self-Organization: a statement of similarities and differences*, 2004.

[10] Freundeskreis Botanischer Garten Aachen e. V. Blütenbiologie - basisinformation und lösungen. http://www.biozac.de/biozac/schule/bluebio.htm.

[11] András Varga et al. The OMNeT++ discrete event simulation system, 2005. http://www.omnetpp.org.

[12] Erich Gamma. *Design patterns*. Addison-Wesley, Boston [u.a.], 2004.

[13] H. Gumm and M. Sommer. *Einführung in die Informatik*. Oldenbourg-Verlag, 5. auflage edition, 2002.

[14] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology. Technical report, IBM T.J. Watson Labs, 2001.

[15] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, June 2006.

[16] Holger Kasinger and Bernhard Bauer. Pollination - a biologically inspired paradigm for self-managing systems. In *Journal of International Transactions on Systems Science and Applications*, volume 2, pages 147 – 156. 2006.

[17] F. Klügl, R. Herrler, and M. Fehler. Sesam: implementation of agent-based simulation using visual programming. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1439–1440, New York, NY, USA, 2006. ACM Press.

[18] Nelson Minar, Rogert Burkhart, Chris Langton, and Manor Askenazi. The swarm simulation system: A toolkit for building multi-agent simulations. Working Papers 96-06-042, Santa Fe Institute, June 1996. available at http://ideas.repec.org/p/wop/safiwp/96-06-042.html.

[19] Christian Müller-Schloer, Christoph von der Malsburg, and Rolf P. Würtz. Organic computing. *Informatik Spektrum*, 27(4):332–336, 2004.

[20] M.J. North, T.R. Howe, N.T. Collier, and J.R. Vos. Repast simphony runtime system. In *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, ANL/DIS-06-1*, Argonne National Laboratory and The University of Chicago, oct 2005.

[21] Rainer Oechsle, Ralph Weires, and Sebastian Lang. Antsim: eine simulationsumgebung zur visualisierung emergenter phänomene am beispiel von ameisenkolonien auf futtersuche. In *SimVis*, pages 227–238, 2004.

[22] Urban Richter, Moez Mnif, Jürgen Branke, Christian Müller-Schloer, and Hartmut Schmeck. Towards a generic observer/controller architecture for organic computing. In Christian Hochberger and Rüdiger Liskowsky, editors, *INFORMATIK 2006 – Informatik für Menschen!*, volume P-93 of *GI-Edition - Lecture Notes in Informatics (LNI)*, pages 112–119. Bonner Köllen Verlag, 2006.

[23] A. Stephan. Varieties of emergence in artificial and natural systems. *Zeitschrift für Naturforschung(A Journal of Biosciences)*, 53c:639–656, 1998.

[24] A. Stephan. *Emergenz. Von der Unvorhersagbarkeit zur Selbstorganisation.* Dresden University Press, Dresden, München, 2. edition, 2005.

[25] Uri Wilensky. NetLogo: Center for connected learning and computer-based modeling, Northwestern University. Evanston, IL, 1999. http://ccl.northwestern.edu/netlogo/.

Anhang A

Benutzung der graphischen Benutzeroberfläche

Wie in Abschnitt 4.4.3 bereits beschrieben, soll dem Benutzer durch eine graphische Oberfläche (GUI) die Bedienung der Simulationsumgebung erleichtert werden. Die GUI des Simulator für selbst-organisierende Systeme führt den Benutzer durch eine Vielzahl an Abfragen, die für die Simulation benötigt werden. Wie in Abbildung 4.7 auf der linken Seite zu sehen ist, kann der Benutzer zu Beginn nur ein Modell auswählen. Da die Simulationsumgebung bisher nur das Bestäubungsmodell unterstützt, hat der Benutzer nur diese Option zur Auswahl. Im nächsten Schritt muss der Benutzer ein Szenario wählen, bevor er sich für eine spezielle Konfiguration des gewählten Szenarios entscheiden kann. Wurde das Modell, das Szenario wie auch die Konfiguration ausgewählt, wird dem Benutzer die Veränderung vorgegebener Werte einiger Parameter ermöglicht. Diese Parameter sind im Folgenden:

Fragrance intensity: Durch die Intensität eines Duftstoffes kann eingestellt werden, wie lang ein Duftstoff in der Umgebung einer Blüte existiert und wie weit er propagiert werden kann. Die Intensität eines Duftstoffes wird einerseits durch jede Propagation (siehe *Beta (β)*) und andererseits durch jede Iteration (siehe *Fragrance decrement*) vermindert. Je höher die Intensität eines Duftstoffes ist, desto mehr Iterationen existiert der Duftstoff und desto weiter kann er propagiert werden.

Intensity increment: Während einer Simulation gibt jede Blüte in bestimmten Abständen Duftstoffe mit einer bestimmten Intensität ab und versucht dadurch Bestäuber anzulocken. Sollte eine Blüte über längere Zeit nicht angeflogen worden sein, so kann die Intensität des neu auszusendenden Duftstoffes um einen Faktor erhöht werden. Dieser Faktor kann durch den Wert *Intensity increment* von dem Benutzer festgelegt werden. Sobald eine Blüte angeflogen wurde, sendet sie wieder Duftstoffe mit der ursprünglichen Intensität aus.

Fragrance decrement: Durch diesen Wert kann der Benutzer einstellen, wie stark die Intensität eines Duftstoffes pro Iteration vermindert werden soll.

Fragrance frequency: Durch diesen Wert kann der Benutzer festlegen, wie oft Duftstoffe von einer Blüte ausgesendet werden.

Reward unit concentration: Die *Reward unit concentration* gibt an, welche Konzentration die Belohnungseinheiten einer Blüte haben. Je höher die Konzentration, desto leichter lässt sich ein Bestäuber anlocken.

Concentration increment: Ähnlich dem Punkt *Intensity increment* kann der Benutzer einstellen, wie stark sich die Konzentration der Belohnungen erhöhen soll, wenn die Blüte längere Zeit von keinem Bestäuber angeflogen wurde.

Vector capacity: Die Kapazität des Bestäubers gibt ab, wie viele Pollen ein Bestäuber maximal geladen haben kann.

Vector search procedure: In der jetzigen Version des Simulator für selbst-organisierende Systeme kann ein Bestäuber nur zufällig in seiner Umgebung suchen. Es ist jedoch vorgesehen, weitere Suchprozeduren eines Bestäubers in das System einzubauen, so dass der Benutzer eine optimale Prozedur auswählen kann.

Flower recreation frequency: Sollte ein Modell so konzipiert sein, dass eine Pflanze nach der Befruchtung einer Blüte diese zerstört und nach einer gewissen Zeit eine neue Blüte der gleichen Art und Gattung produziert, so kann der Benutzer über die *Flower recreation Frequency* die Anzahl der Iterationen angeben, die eine Pflanze warten muss, bis sie eine neue Blüte erzeugen darf.

Beta (β): Durch diesen Wert kann die Abnahme der Intensität eines Duftstoffes durch den Benutzer leicht verändert werden. Wird ein Duftstoff von einem Feld auf ein Nachbarfeld propagiert, so nimmt dessen Intensität ab. Die Menge der Abnahme richtet sich nach dem Nachbarfeld, d.h. wurde der Duftstoff horizontal oder vertikal propagiert, so vermindert sich der Wert um '1'. Wurde er hingegen diagonal propagiert, so vermindert sich der Wert um $\sqrt{2}$. Durch den Wert β kann der Benutzer die Verminderung steuern. Demnach wird ein horizontal bzw. vertikal propagierter Duftstoff um 1^{β}, ein diagonal propagierter Duftstoff um $\sqrt{2}^{\beta}$ vermindert. Der Wert β muss in dem Intervall]0,1] liegen.

Treshold: Der Schwellwert des Systems gibt den Prozentsatz an, mit dem ein Bestäuber dem für ihn besten Duft folgt. Dieser Wert muss in dem Intervall [0,100]% liegen. Stuft ein Bestäuber einen gerochenen Duftstoff als den momentan besten Duftstoff ein, so folgt er diesem in Abhängigkeit zu dem Schwellwert. Hat der Benutzer beispielsweise 60% angegeben, so folgt der Bestäuber dem ausgewählten Duftstoff nur mit einer Wahrscheinlichkeit von 60%. Im Falle der 40% folgt der Bestäuber nicht dem Besten, aber der besten aggregierten Menge an Duftstoffen.

Wenn der Benutzer mit den Einstellungen seiner Parameter zufrieden ist, kann er das ausgewählte Simulationsexperiment laden. Dieses wird nun auf der Oberfläche angezeigt und kann durch einen Klick auf den "start"-Knopf simulieren werden. Der Benutzer kann während der Simulation die einzelnen Zustände des Experiments auf der Oberfläche mit verfolgen und bei Bedarf die Simulation unterbrechen oder stoppen. Soll eine neue Konfiguration simuliert werden, so kann der Benutzer durch einen Klick auf den "clear"-Knopf das bisherige Experiment löschen und neue Einstellungen vornehmen.

Anhang B

Ergänzende Evaluationsergebnisse

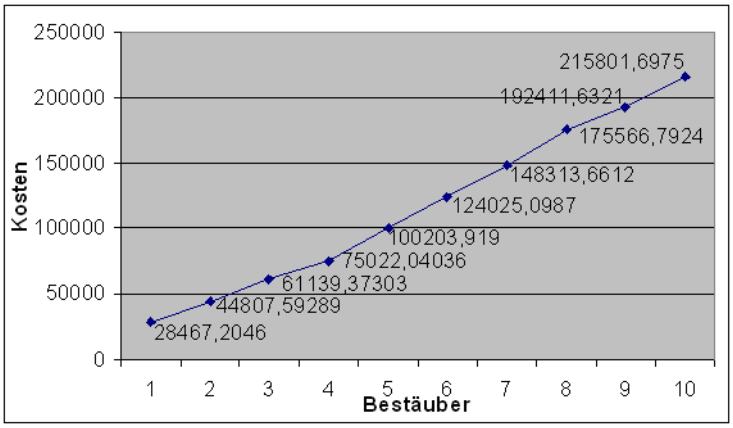

Abbildung B.1: Genaue Kosten des einfachen Simulationsmodells bei steigender Anzahl an Bestäubern.

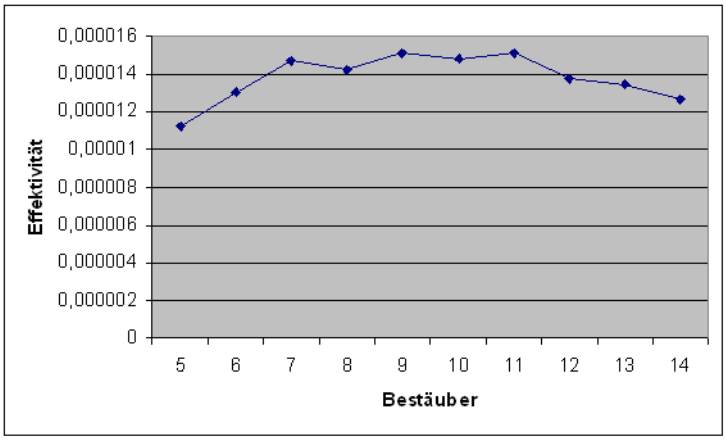

Abbildung B.2: Effektivität des einfachen Simulationsmodells mit 4 Blüten in Abhängigkeit von der Anzahl an Bestäubern.

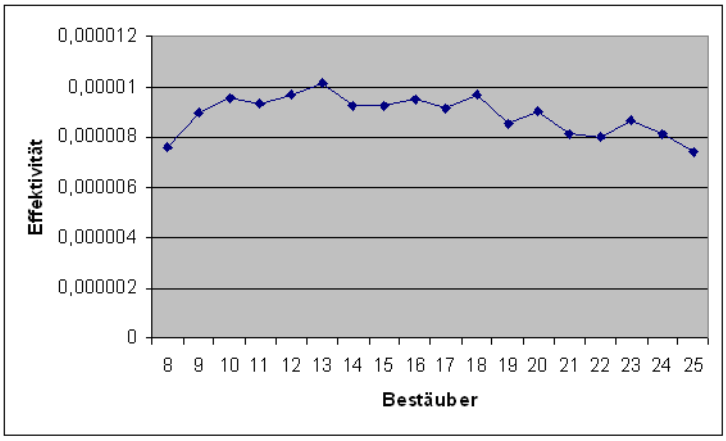

Abbildung B.3: Effektivität des einfachen Simulationsmodells mit 6 Blüten in Abhängigkeit von der Anzahl an Bestäubern.

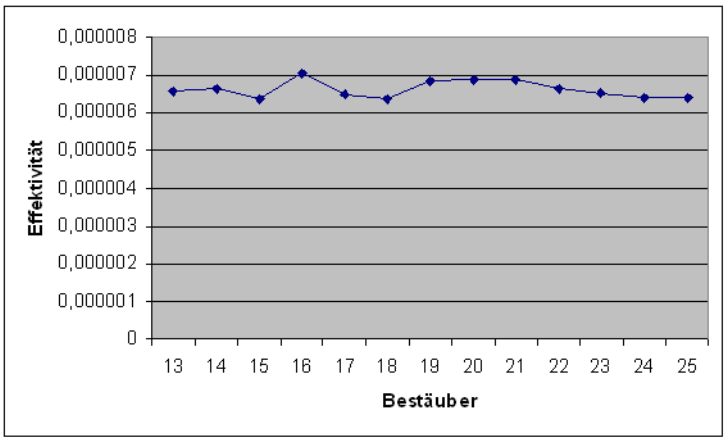

Abbildung B.4: Effektivität des einfachen Simulationsmodells mit 8 Blüten in Abhängigkeit von der Anzahl an Bestäubern.

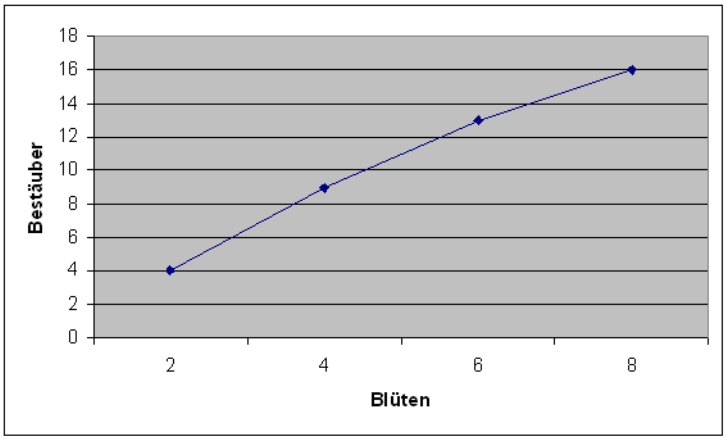

Abbildung B.5: Optimale Blüten-Bestäuber-Paare basierend auf dem einfachen Simulationsmodell mit 2,4,6 und 8 Blüten.